Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Discrete Applied Mathematics 129 (2003) 233–262
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Pushdown–reduce: an algorithm for connectivity augmentation and poset covering problems

András A. Benczúr[a,b,*,1]

[a] *Computer and Automation Institute, Hungarian Academy of Sciences, Eötvös University, Budapest, Hungary*
[b] *Department of Operations Research, Eötvös University, Budapest, Hungary*

Received 17 January 2001; received in revised form 30 April 2002; accepted 1 July 2002

Abstract

In their seminal paper, Frank and Jordán show that a large class of optimization problems including certain directed edge augmentation ones fall into the class of *covering supermodular functions over pairs of sets*. They also give an algorithm for such problems, however, that relies on the ellipsoid method.

Our main result is a combinatorial algorithm for the restricted covering problem when the supermodular function is 0–1 valued; the problem includes directed vertex or $S-T$ connectivity augmentation by one. Our algorithm uses an approach completely different from that of an independent recent result of Frank. It finds covers of partially ordered sets that satisfy natural abstract properties slightly extending those in Frank and Jordán. The algorithm resembles primal–dual augmenting path algorithms: For an initial (possibly greedy) cover the algorithm searches for witnesses for the necessity of each element in the cover. If no two witness have a common cover, the solution is optimal. As long as this is not the case, the witnesses are gradually exchanged by smaller ones (PUSHDOWN step). Each witness change defines an appropriate change in the solution; these changes are finally unwound in a shortest path manner to obtain a solution of size one less (REDUCE step).

1. Introduction

Edge connectivity augmentation problems form a subclass of survivable network design [14], where one is interested in the minimum number of edges needed to be

* Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, Hungary.
E-mail address: benczur@cs.elte.hu (A.A. Benczúr).
URL: http://www.cs.elte.hu/∼benczur
[1] Supported from grants OTKA T-30132 and T-29772; NWO-OTKA N-34040; AKP 104024.

0166-218X/03/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
PII: S0166-218X(02)00570-X

added to a graph to satisfy certain connectivity prescriptions. Algorithms for various augmentation problems have a large and expanding literature [1,3,4,6,10,13,18,22,23] and many others.

In this paper we give a combinatorial algorithm for a special class of augmentation problems introduced by Frank and Jordán [10] as *covering supermodular functions over pairs of sets*. The problem of finding the minimum number of directed edges that make a directed graph G k-vertex-connected is the central example. We may consider all cuts of G with less than k vertices as set pairs (X, Y) of the vertex set where X is the source and Y is the sink side of the cut (recall the graph is directed). Let

$$p(X, Y) = \max\{0, k - (|V| - |X| - |Y|)\}$$

denote the number of vertices "missing" for a k-connected graph; the graph becomes k-connected iff for all (X, Y) we add at least $p(X, Y)$ edges that lead from X to Y.

The above demand function p satisfies the following crossing supermodular property: whenever $X \cap X' \neq \emptyset$, $Y \cap Y' \neq \emptyset$ and $p(X, Y) > 0$, $p(X', Y') > 0$,

$$p(X \cap X', Y \cup Y') + p(X \cup X', Y \cap Y') \geqslant p(X, Y) + p(X', Y').$$

When specialized for $p \leqslant 1$, this means that the collection of pairs with positive demand $p(X, Y)$ are closed under the skew intersection/union operation as in the left-hand side of the inequality.

In the above formulation we restrict attention to problems where the demand function p is 0–1 valued, corresponding to increasing directed vertex connectivity by one. Another example [10] is increasing directed $S-T$ connectivity by one: For two possibly overlapping vertex sets S and T, the $S - T$ connectivity is the maximum number of directed edge-disjoint paths that connect pairs of vertices with head in S and tail in T. Connectivity is increased by one iff we add at least one edge for all pairs $X \subseteq S$, $Y \subseteq T$ such that the number of directed edges with head in Y and tail in X is minimum over all possible pairs of such subsets.

Yet another remarkable problem that falls into the class of our cover problems is Győri's rectangle cover problem [16]. In an equivalent formulation the problem asks for the minimum number of subpaths $\mathscr{I}$ of a ground path such that each path of a system of subpaths $\mathscr{P}$ arises as the union of some paths in $\mathscr{I}$. Here the reduction to systems of set pairs is to take individual edges of subpaths in $\mathscr{P}$ and define set pairs by taking the vertices of the subpath left and right from the selected edge. For this problem combinatorial algorithms are known both based on the Frank–Jordán algorithm [8,7,2] and on direct combinatorial proofs [11,21,20].

The powerful result of Frank and Jordán [10] gives a min–max result for arbitrary valued supermodular demands p. However the algorithm they give uses the *ellipsoid method* [15]. The search for combinatorial algorithms continues with previously only minor success in handling certain special problems [7,8]. Independent of our result, a completely different algorithm for the 0–1 demand case is recently given by Frank [9].

In the heart of most results related to covering problems over set pairs, we find Dilworth's theorem stating that the minimum number of chains that cover a partially ordered set is equal to the maximum number of pairwise incomparable elements of the set. Both the non-combinatorial algorithm [10] and certain combinatorial ones [7,8,2] including the recent result of Frank [9] for covering set pairs start with a reduction to chain covers as in Dilworth's theorem.

Our new combinatorial algorithm circumvents the reduction to Dilworth's theorem. Instead we show how the slightly modified folklore Dilworth algorithm extends to the more general problem of covering set pairs. It is easy to show that for each directed edge (x, y) there is a unique set pair (X, Y) with X minimum and Y maximum and another (X', Y') with Y' minimum and X' maximum with $x \in X, X'$ and $y \in Y, Y'$. All other (X'', Y'') satisfy this property iff $X \subseteq X'' \subseteq X'$ and $Y \supseteq Y'' \supseteq Y'$. Thus if we define a partially order with the above "skew" containment relation, we get the problem of covering a partially ordered set by intervals, a direct generalization of Dilworth's problem.

We construct an optimum interval cover by starting with an arbitrary greedy cover and gradually improve it in a primal–dual augmenting path manner that mimics standard Dilworth algorithms. Algorithms for Dilworth's theorem are based on a reduction to the bipartite matching problem [12]. When we unfold the reduction of Dilworth's theorem to bipartite matchings, we find that the classical alternating path matching algorithm translates into an algorithm that (i) maintains one element for each chain as a candidate dual optimum; (ii) terminates with optimum if these elements are pairwise incomparable; and (iii) otherwise use these elements to guard exchanges in chain parts such that one of the chains eventually becomes unnecessary for the cover. Such a direct Dilworth algorithm is described in Frank [5]. We remark that the current best bipartite matching algorithm is given in [17] and for Dilworth's problem in [2].

The paper is organized as follows. In Section 2 we describe a direct algorithm for a restricted version of Dilworth's theorem. While this algorithm itself is of less interest partly since it is only a slight modification over that of Frank [5] and partly because of the hardness of its proof, this algorithm forms the base of the main result of Section 3. Having read the main ideas preceding Section 2.1, one may read the paper by jumping directly to Section 3. However, Section 2 contains a large amount of explanations intended to illustrate the ideas in the main algorithm and Section 3 itself only gives the details necessary to give the algorithm and prove its correctness.

2. The algorithm for Dilworth's theorem

A partially ordered set or *poset* for short is a set $\mathscr{P}$ with a binary relation "$<$" such that $x < y$ implies $x/ > y$ and $x \neq y$, where "$=$" means the identity of set elements. From relation $<$ relation $\leqslant$ is obtained in the obvious way. Unlike in a total order we do not require one of $x \leqslant y$ or $y \leqslant x$ to hold; pairs of elements $x \not\leqslant y$ and $y \not\leqslant x$ are called *incomparable* or *independent*. Element x is a *predecessor* of another y, denoted $x \prec y$, if $x < y$ but $x < z < y$ for no $z \in \mathscr{P}$.

Dilworth's theorem states that the minimum number of chains (increasing sequences of poset elements) that cover a poset in the natural sense defined below is equal to the maximum size of a pairwise independent system, cf. [12]. We use *Dilworth's problem* as the name of the task to find the minimum and maximum in question.

Definition 2.1. Given a poset $\mathscr{P}$, we say that a collection $\mathscr{I} = \{[m_i, M_i] : i \leqslant k\}$ of pairs of minimal and maximal poset elements *cover the poset* if for all $x \in \mathscr{P}$ we have $m_i \leqslant x \leqslant M_i$ for some $i \leqslant k$.

In this section we give an algorithm for Dilworth's problem for special posets satisfying the definition below. The algorithm could with some care be modified to work for general posets as well. However, our algorithm is not significantly different from the one obtained by unfolding the standard reduction of Dilworth's problem to bipartite matchings as a direct augmenting path algorithm over the poset. Such an algorithm is described for instance in [5]. We emphasize that the sole purpose of the algorithm in this section is a motivation for the main algorithm of Section 3.

Definition 2.2. We say that a poset $\mathscr{P}$ satisfies the *unique path* property if

$$\text{for all } x, y, u, v \in \mathscr{P} \quad \text{with } x \leqslant u \leqslant y \text{ and } x \leqslant v \leqslant y \text{ we have } u \leqslant v \text{ or } v \leqslant u.$$

By this definition maximal chains have the form $\{x : m \leqslant x \leqslant M\}$ where m is a minimal and M is a maximal poset element. Hence chains are identical to intervals $[m, M]$. The notation $[m, M]$ will interchangeably denote the interval or the pair of minimal and maximal elements.

Definition 2.3. Given a cover $\mathscr{I} = \{[m_i, M_i] : i \leqslant k\}$, a poset element x is *tight* if there is a unique $j \leqslant k$ with $m_j \leqslant x \leqslant M_j$.

The overall structure of our algorithm is given in Algorithm PUSHDOWN–REDUCE. The pseudocode below forms the base of both our Dilworth solution and the more general main result of Section 3. We start with a (possibly greedy) chain cover and a tight element of each chain in the solution that can be considered a witness for the need of that particular chain. If the set of witnesses is pairwise independent, we are in optimality: any set of poset elements with size more than the number of chains contains comparable pairs by the pigeonhole principle while any set of chains of smaller size avoid at least one of the witnesses.

In general however, witnesses will not directly prove the optimality of the solution. Then they will gradually be exchanged by smaller ones (PUSHDOWN step). Each witness change defines an appropriate change in the cover without changing its cardinality; these changes are recorded. Algorithm PUSHDOWN–REDUCE either terminates with witnesses that give proof of optimality, or else it terminates with one of the witnesses vanishing, in a sense defined within the PUSHDOWN step. In this second case if we follow the recorded changes from the vanished witness backward, we may finally discard one element of the solution (REDUCE step).

Algorithm PUSHDOWN–REDUCE(poset $\mathscr{P}$, cover $\mathscr{I}$)
  for $j = 1,\dots,k$ **do**
    $u_j^{(1)} \leftarrow \max\{x\colon x \text{ is tight with } m_j \leqslant x \leqslant M_j\}$
    if $u_j^{(1)}$ does not exist **then**
      exit with reduced cover $\{[m_i, M_i] : i = 1,\dots,j-1, j+1,\dots,k\}$
  for $t = 1, 2, \dots$ **do**
    if the $u_i^{(t)}$ are pairwise independent **then**
      exit with maximum independent system found
  for $j = 1,\dots,k$ **do**
    $u_j^{(t+1)} \leftarrow$ PUSHDOWN(j, $\{u_i^{(t)}\colon i \leqslant k\}$)

For the ease of notation we define the poset $\mathscr{P}$, the cover $\mathscr{I}$, variables $u_i^{(t)}$ and the value of t global; i.e. we assume the corresponding values are available whenever they are needed in subroutines PUSHDOWN and REDUCE. We also make it possible to exit before loops and subroutine calls are terminated; the algorithm itself, for instance, will terminate by a call to Procedure REDUCE within a certain call to Subroutine PUSHDOWN.

2.1. The PUSHDOWN step

In the next two subsection we give the two main steps of the Dilworth algorithm. The uninterested reader may now skip to Section 3.

The purpose of the PUSHDOWN step is to gradually replace witnesses u_i that are tight with $m_i \leqslant u_i \leqslant M_i$ by smaller ones until we find no tight element below certain u_j. In that case we proceed with rewinding the witness changes to reduce the cover size by one with the REDUCE step of the next subsection.

The main idea of a witness change is that unless the set of u_i are pairwise independent, we find pairs $u_{j'} \leqslant u_j$. Now if we replace $[m_{j'}, M_{j'}] \in \mathscr{I}$ by $[m_{j'}, M_j]$ to obtain another cover $\tilde{\mathscr{I}}$, element u_j will no longer be tight and we may proceed to a smaller u_j. Notice that in $\tilde{\mathscr{I}}$ we may loose the cover of certain $u_{j'} \leqslant x \leqslant M_{j'}$; this flaw is removed in the next section when a sequence of replacements will be constructed that fit together such that $\tilde{\mathscr{I}}$ remains a cover.

Consider a pair $u_{j'} \leqslant u_j$. When $\tilde{\mathscr{I}}$ as above is defined, no element $m_j \leqslant x \leqslant M_j$ remains tight with $m_{j'} \leqslant x$. The next definition captures the minimum element that is not tight and the maximum that may remain tight after such a change in the cover.

Definition 2.4. If $m \neq m'$ are both smaller than or equal to M, then both

$$u(m, m', M) = \max\{x\colon m \leqslant x \leqslant M \text{ and } x \not\geqslant m'\},$$

$$U(m, m', M) = \min\{x\colon m \leqslant x \leqslant M \text{ and } x \geqslant m'\}$$

exist by the unique path property. Given $U(m,m',M)$, $u(m,m',M)$ is the unique element $u(m,m',M) \prec U(m,m',M)$ between m_j and M_j. For $m = m_j$, $M = M_j$, $m' = m_{j'}$ we use

$$u_{jj'} = u(m_j, m_{j'}, M_j) \quad \text{and} \quad U_{jj'} = U(m_j, m_{j'}, M_j)$$

whenever this notation causes no ambiguity.

Procedure PUSHDOWN(j, $\{u_i\}$)
  $u \leftarrow u_j$
  while $\exists j' \neq j$ with $u_{j'} \leqslant u$ **or** $m_{j'} \leqslant u \leqslant M_{j'}$ **do**
    if $m_{j'} = m_j$ **then**
      exit with reduced cover REDUCE(j)
    else
      $u \leftarrow u_{jj'}$
  return u

Now we are ready to give Procedure PUSHDOWN. We proceed by selecting a decreasing sequence of poset elements $u = u_{ji}$. By definition all these elements satisfy $m_j \leqslant u_{ji} \leqslant M_j$, thus when the final value of u is returned to the calling Algorithm PUSHDOWN–REDUCE, we get

$$m_j \leqslant \ldots \leqslant u_j^{(t)} \leqslant u_j^{(t-1)} \leqslant \ldots \leqslant u_j^{(1)} \leqslant M_j. \tag{1}$$

2.2. ALTERNATE-PUSHDOWN *and initialization procedures*

We give alternate procedures for PUSHDOWN and the initialization of $u_j^{(1)}$. The advantages or disadvantages of the two versions of the procedures mainly depend on the actual implementation of poset oracles that, for example, find maximum tight elements or $u(m,m',M)$. The discussion in Section 3 will depend on the original form of the initialization but the new form of PUSHDOWN, a pair of procedures similar to one another.

Procedure PUSHDOWN may make an extra step in addition to changing $u_j \leftarrow u_{jj'}$ if $u_{j'} \leqslant u_j$ for some $j' \neq j$: the condition $m_{j'} \leqslant u_j \leqslant M_{j'}$ means we may also make such a step if $u_{j'} \geqslant u_j$. This is no surprise, since u_j is not tight in that case. We may say one type of step prepares the actual changes in the chain cover while the other type selects tight elements.

Next we give Procedure ALTERNATE-PUSHDOWN where we separate the steps for preparing cover changes and selecting tight elements in Procedure PUSHDOWN more clearly, in two separate minimization and maximization lines. This version of the procedure will then form the base of the more general algorithm in Section 3.

Procedure ALTERNATE-PUSHDOWN($j, \{u_i\}$)
  $u' \leftarrow \min\{u_j;\ u_{jj'} : u_{j'} \leqslant u_j$ and $j \neq j'\}$
    /nonexistent if $m_j = m_{j'}$ for at least one pair j, j'/
  $u \leftarrow \max\{x : x$ is tight with $m_j \leqslant x \leqslant u'\}$
    /nonexistent if no such x is tight/
  if u does not exist **then**
    exit with reduced cover REDUCE(j)
  return u

In this subsection we also rephrase the initialization step

$$u_j^{(1)} \leftarrow \max\{x : x \text{ is tight with } m_j \leqslant x \leqslant M_j\}.$$

In its present form initialization is identical with the second step of Procedure ALTERNATE-PUSHDOWN. We give Procedure ALTERNATE-INIT where a sequential replacement by elements $u_{jj'}$ similar to Procedure PUSHDOWN is used; the proof that the outcome is equal to $u_j^{(1)}$ is identical to that of the lemma below for PUSHDOWN.

It is almost immediate that Procedures PUSHDOWN and ALTERNATE-PUSHDOWN produce identical output, since apparently both algorithms replace u by $u_{jj'}$ for the same set of j'. This could however not necessarily be true since Procedure ALTERNATE-PUSHDOWN may in the definition of u' select j' with $u_{j'} \leqslant u_j$ while for the decreasing value of u in Procedure PUSHDOWN $u_{j'} \leqslant u < u_j$ may no longer hold. In the next proof this is the main difficulty we face; we also have to carefully handle the case when either of the procedures could terminate by exiting to Procedure REDUCE.

Lemma 2.5. *Procedure* PUSHDOWN *and Procedure* ALTERNATE-PUSHDOWN *produce identical output.*

Proof. To simplify the proof, let us augment chain $[m_j, M_j]$ by an additional element $-\infty$ below m_j. Instead of exiting to Procedure REDUCE, we may say both algorithms select $-\infty$ as return value. Indeed, in both procedures we may redefine $u_{jj'}$ for $m_j = m_{j'}$ to be $-\infty$; and in Procedure ALTERNATE-PUSHDOWN we always have $-\infty$ as a (tight) element in the set to be maximized.

In the above augmented poset let u^* denote the return value of Procedure PUSHDOWN and $\tilde{u}^*$ of Procedure ALTERNATE-PUSHDOWN. First we show $u^* \geqslant u^*$; notice the two elements are comparable since they both belong to chain $[-\infty, M_j]$. Since in Procedure PUSHDOWN we get u^* by repeatedly changing u to a smaller $u_{jj'}$, we assume by contradiction that in some step u is moved below $\tilde{u}^*$. In other words, this means that the value of u before executing $u \leftarrow u_{jj'}$ satisfies $u \geqslant \tilde{u}^* > u_{jj'}$. Now we distinguish two cases depending on which part of the **or** operator is true at this execution.

- If $u_{j'} \leqslant u$, then by $u \leqslant u_j$ we get that j' is included in the minimization of Procedure ALTERNATE-PUSHDOWN, thus $\tilde{u}^* \leqslant u_{jj'}$.

- And if $m_{j'} \leqslant u \leqslant M_{j'}$, then by $m_{j'} \leq U_{jj'}$, $u_{jj'} \prec U_{jj'}$ and $\tilde{u}^* > u_{jj'}$ we get $m_{j'} \leqslant U_{jj'} \leqslant \tilde{u}^* \leqslant u \leqslant M_{j'}$, contradicting that $\tilde{u}^*$ is selected to be a tight element.

Now we prove $u^* \leqslant \tilde{u}^*$. Let u' denote the result of minimization in Procedure ALTERNATE-PUSHDOWN; $u' = u_{jj'}$ for some $u_{j'} \leqslant u_j$. We cannot have $u_{j'} \leqslant u^*$, since then the *while* loop of Procedure PUSHDOWN can be executed with j' and u^* can be replaced by a smaller poset element. Assume by contradiction that $u^* > u'$; by $u' = u_{jj'}$ this implies $u^* \geqslant U_{jj'}$. We have obtained $m_{j'} \leqslant u_{j'} \leqslant u_j \leqslant M_j$ and $m_{j'} \leqslant U_{jj'} \leqslant u^* \leqslant M_j$; by the unique path property these imply u^* and $u_{j'}$ are comparable and then $u^* \leqslant u_{j'}$. Now the *while* loop of Procedure PUSHDOWN can be executed again with j' since $m_{j'} \leqslant u^* \leqslant u_{j'} \leqslant M_{j'}$.

We proved $u^* \leqslant u'$; we are done if we show u^* is tight. This follows since if $m_{j'} \leqslant u^* \leqslant M_{j'}$ for some $j' \neq j$, then the *while* loop of Procedure PUSHDOWN could be executed and u^* could be replaced by a smaller poset element. □

Procedure ALTERNATE-INIT(j)
  $u_j = M_j$
  while $\exists j' \neq j$ with $m_{j'} \leqslant u_j \leqslant M_{j'}$ **do**
    if $m_{j'} = m_j$ **then**
      exit with reduced cover REDUCE(j)
    else
      $u_j \leftarrow u_{jj'}$
  return u_j

Finally we remark that the selection of a tight element in Line 2 could be omitted from Procedure ALTERNATE-PUSHDOWN; in the same way we could just let $u_j^{(1)} = M_j$ in Procedure ALTERNATE-INIT. This would, of course, produce different sequences $u_i^{(t)}$; however, Algorithm PUSHDOWN–REDUCE would still be correct. The next proof can be modified at certain places in order to show the correctness of the modified algorithm. While this modified algorithm is somewhat simpler, it might be less efficient, and, more importantly, the more general algorithm of Section 3 relies on the values $u_i^{(t)}$ in the form produced by the original Procedure PUSHDOWN.

2.3. The REDUCE step

The REDUCE step traverses all changes backward from the last execution of Procedure PUSHDOWN with $t = t^*$. One element $u_{j_s}^{(t)}$ is selected for each $t = t^*, t^* - 1, \ldots, 1$ where $s = t^* + 1 - t$ is used to ease notation. These poset elements constitute a "shortest augmenting path": when replacing each chain endpoint m_{j_s} by the endpoint $m_{j_{s-1}}$ of the preceding element, the first chain $[m_{j_1}, M_{j_1}]$ can be removed from the cover.

Procedure REDUCE(j)
  $j_1 \leftarrow j$; $t^* \leftarrow t$;
  for $t = t^*, \ldots, 1$; $s = 1, \ldots, t^*$ **do**
    $j_{s+1} \leftarrow$value with $u_{j_s j_{s+1}}$ minimum in $\{u_{j_s j} : u_j^{(t)} \leqslant u_{j_s}^{(t)}\}$
    $m_{j_s} \leftarrow m_{j_{s+1}}$
  return $\{[m_i, M_i] : i \leqslant k,\ i \neq j_{t^*+1}\}$

The idea behind the choice of $u_{j_s}^{(t)}$ is to leave no tight element below $u_{j_1}^{(t^*)}$, above $u_{j_{t^*+1}}^{(1)}$, and between the parts of the chain $[m_{j_s}, M_{j_s}]$ contained by the two new chains $[m_{j_{s+1}}, M_{j_s}]$ and $[m_{j_s}, M_{j_{s-1}}]$. The second requirement is satisfied by choosing the value of j that causes the call to Procedure REDUCE for j_1. Then in subsequent steps we always select j_{s+1} such that in iteration $t = t^* + 1 - s$ the first line in Procedure ALTERNATE-PUSHDOWN lets $u' = u_{j_s j_{s+1}}$ (see Fig. 1 for the case $t^* = 2$).

While the actual proof of correctness is apparently harder than the above simple reasoning, for a slightly modified algorithm (not including the choice of a tight u in Procedure ALTERNATE-PUSHDOWN) one could give a much simpler proof along these lines as in [5]. However, our main purpose here is to give a proof that motivates the algorithm of the next section as much as possible and there we require an algorithm of the current form.

Theorem 2.6. *If Procedure* REDUCE *is called by Algorithm* PUSHDOWN–REDUCE, *then the output of the procedure is a cover of the poset.*

In the first iteration in Procedure REDUCE we replace the cover $\mathscr{I}$ by changing m_{j_1} to $\tilde{m}_{j_1} = m_{j_2}$ to get

$$\tilde{\mathscr{I}} = \{[m_i, M_i] : i \neq j_1\} \cup \{[m_{j_2}, M_{j_1}]\}. \tag{2}$$

We begin the proof by showing that the resulting set is also a cover.

Lemma 2.7. *Let the first iteration in Procedure* REDUCE *replace $\mathscr{I}$ by $\tilde{\mathscr{I}}$ as in* (2). *Then $\tilde{\mathscr{I}}$ is a cover.*

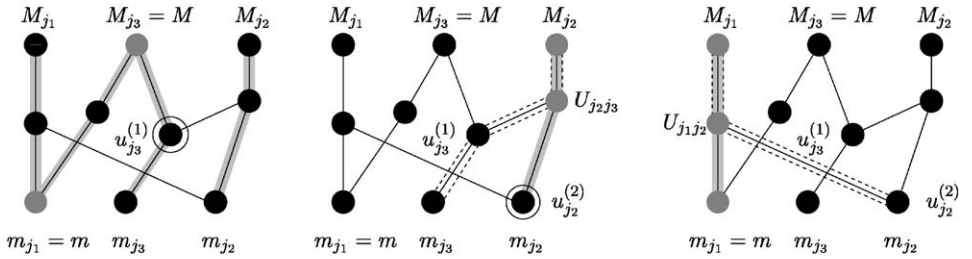
Proof. Let $u = u_{j_1}^{(t^*)} = u_{j_1 j_2}$ and $U = U_{j_1 j_2}$. By $u \prec U$ we get that $\tilde{\mathscr{I}}$ is a cover, since

$$m_{j_1} \leqslant x \leqslant u \quad \text{implies } x \text{ is not tight in } \mathscr{I}, \text{ thus covered in } \tilde{\mathscr{I}}; \tag{3}$$

$$U \leqslant x \leqslant M_{j_1} \quad \text{implies } \tilde{m}_{j_1} = m_{j_2} \leqslant x \leqslant M_{j_1}, \tag{4}$$

where (3) follows by the condition to call Procedure REDUCE from Procedure PUSHDOWN and (4) follows since $m_{j_2} \leqslant U_{j_1 j_2} = U$. □

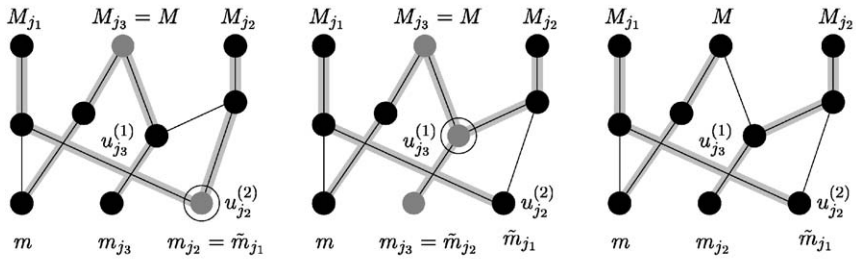
Proof of Theorem 2.6 *for small* t^*. If $t^* = 1$, then Procedure REDUCE terminates after the first iteration. In this case $u_{j_2}^{(1)}$ is the largest tight element x with $m_{j_2} \leqslant x \leqslant M_{j_2}$.



(a) Start of Algorithm: Four chains cover the poset; shaded poset nodes are non-tight; $u^{(1)}_{j_3} \in [m_{j_3}, M_{j_3}]$ is maximum tight.

(b) Step 2: Since $u^{(1)}_{j_3} \leq u^{(1)}_{j_2} = M_{j_2}$, we define $u^{(2)}_{j_2} \prec U_{j_2j_3}$. Dashed chain indicates the possible new covering chain with two new non-tight shaded nodes.

(c) Step 3: Since $u^{(2)}_{j_2} \leq u^{(2)}_{j_1} = M_{j_1}$ and there is no tight element below $U_{j_1j_2}$, REDUCE is called.



(d) REDUCE Step 1: $t = t^* = 2$ and $s = t^* + 1 - t = 1$. The new set of chains and new non-tight elements are shaded.

(e) REDUCE Step 2: Proceeds $t = 1$ and $s = 2$ by selecting j_{s+1} for the $u_{j_sj_{s+1}}$ circled.

(f) End of Algorithm: the new poset with three covering chains. Chain $[m_{j_3}, M_{j_3}]$ is removed.

Fig. 1. Illustration of Procedure REDUCE for $t^* = 2$. (a) Start of Algorithm: Four chains cover the poset; shaded poset nodes are non-tight; $u^{(1)}_{j_3} \in [m_{j_3}, M_{j_3}]$ is maximum tight. (b) Step 2: Since $u^{(1)}_{j_3} \leqslant u^{(1)}_{j_2} = M_{j_2}$, we define $u^{(2)}_{j_2} \prec U_{j_2j_3}$. Dashed chain indicates the possible new covering chain with two new non-tight shaded nodes. (c) Step 3: Since $u^{(2)}_{j_2} \leqslant u^{(2)}_{j_1} = M_{j_1}$ and there is no tight element below $U_{j_1j_2}$, REDUCE is called. (d) REDUCE Step 1: $t = t^* = 2$ and $s = t^* + 1 - t = 1$. The new set of chains and new non-tight elements are shaded. (e) REDUCE Step 2: Proceeds $t = 1$ and $s = 2$ by selecting j_{s+1} for the $u_{j_sj_{s+1}}$ circled. (f) End of Algorithm: the new poset with three covering chains. Chain $[m_{j_3}, M_{j_3}]$ is removed.

However, in $\tilde{\mathscr{J}}$ all elements $m_{j_2} \leqslant x \leqslant u^{(1)}_{j_2}$ are covered by both $[m_{j_2}, M_{j_2}]$ and $[\tilde{m}_{j_1}, M_{j_1}] = [m_{j_2}, M_{j_1}]$ by $M_{j_1} \geqslant u^{(1)}_{j_1} \geqslant u^{(1)}_{j_2}$. Thus we have a reduced cover if we remove the superfluous $[m_{j_2}, M_{j_2}]$ as in the last line of Procedure REDUCE.

For larger values of t^* we use induction. To illustrate the main idea of the proof, we show how to handle $t^* = 2$ first. In that case we replace $[m_{j_1}, M_{j_1}]$, $[m_{j_2}, M_{j_2}]$ and $[m_{j_3}, M_{j_3}]$ in $\mathscr{J}$ by $[m_{j_2}, M_{j_1}]$ and $[m_{j_3}, M_{j_2}]$. The chains are changed in

three steps:

- First $[m_{j_1}, M_{j_1}]$ is replaced by $[m_{j_2}, M_{j_1}]$ by removing one cover of

  $$m_{j_1} \leqslant x \leqslant u_{j_1 j_2};$$

  these x are not tight by definition—in addition to $[m_{j_1}, M_{j_1}]$ they may be covered by $[m_j, M_j]$ for any $j \neq j_2$, in particular by $j = j_3$, a chain that is removed from the new chain cover.
- Next $[m_{j_2}, M_{j_2}]$ is replaced by $[m_{j_3}, M_{j_2}]$, removing a covering chain of

  $$m_{j_2} < x \leqslant u_{j_2 j_3}.$$

  However all $m_{j_2} \leqslant x \leqslant u^{(2)}_{j_2} \leqslant M_{j_1}$ have the new chain $[m_{j_2}, M_{j_1}]$ covering them. Thus the only elements with possibly no cover are $u^{(2)}_{j_2} < x \leqslant u_{j_2 j_3}$ with $x \notin M_{j_1}$.
  By the choice of j_3 in Procedure REDUCE and of $u' = u_{j_2 j_3}$ in Procedure ALTERNATE-PUSHDOWN these elements are not tight in the original cover. They are thus covered by $[m_{j_2}, M_{j_2}]$ and some $[m_j, M_j]$ in addition. Since $x \leqslant u_{j_2 j_3}$ and $x \notin M_{j_1}$, this cover j is neither j_1 nor j_3, hence it is a chain still present in the new cover.
- Finally we remove $[m_{j_3}, M_{j_3}]$. All elements

  $$m_{j_3} \leqslant x \leqslant u^{(1)}_{j_3} \leqslant u^{(1)}_{j_2} \leqslant M_{j_2}$$

  are covered by the new chain $[m_{j_3}, M_{j_2}]$. Let us consider $u^{(1)}_{j_3} < x \leqslant M_{j_3}$ with $x \notin M_{j_2}$; by Procedure INIT no such x is tight in the initial cover, thus $m_j \leqslant x \leqslant M_j$ for some $j \neq j_3, j_2$. Thus they remain covered unless $j = j_1$.

By enumerating the three cases above, we saw all x are covered in the new set of chains except for either

$$m_{j_1} \leqslant x \leqslant u_{j_1 j_2} \quad \text{with } m_{j_3} \leqslant x \leqslant M_{j_3}, \tag{5}$$

$$u^{(1)}_{j_3} < x \leqslant M_{j_3} \quad \text{with } m_{j_1} \leqslant x \leqslant M_{j_1}. \tag{6}$$

In both cases x is contained by both chains j_1 and j_3, thus the chains overlap and $U_{j_1 j_3}$ exists. In addition if x in either case remains uncovered, the overlapping parts of the chains include the changing regions. Since adding $[m_{j_3}, M_{j_2}]$ and removing $[m_{j_3}, M_{j_3}]$ affects elements above $u^{(1)}_{j_3}$, the conditions of the next lemma will be met whenever we may have x uncovered in the proof below. In addition to being hence technically useful, the lemma can also be interpreted as j_3 cannot be used to define $u^{(t)}_{j_2}$ (and $j_{t'}$ to $u^{(t)}_{j''_t}$ for $t' < t''$) in Procedure ALTERNATE-PUSHDOWN.

Lemma 2.8. *If $u^{(1)}_{j_3} \leqslant M_{j_1}$, then $U_{j_1 j_3}$ exists and $U_{j_1 j_2} < U_{j_1 j_3}$.*

Proof. The existence of $U_{j_1 j_3}$ follows by $m_{j_3} \leqslant u^{(1)}_{j_3} \leqslant M_{j_1}$. Since $u^{(1)}_{j_3}$ is tight, we may not have $m_{j_1} \leqslant u^{(1)}_{j_3}$, implying $u^{(1)}_{j_3} < U_{j_1 j_3}$. Now we prove by contradiction:

$$u^{(1)}_{j_3} < U_{j_1 j_3} \leqslant U_{j_1 j_2} \leqslant u^{(2)}_{j_1} \leqslant u^{(1)}_{j_1},$$

where the third inequality follows by the choice of j_2 in Procedure REDUCE. Now we are in contradiction with the definition of $u^{(2)}_{j_1}$ in Procedure ALTERNATE-PUSHDOWN since $u^{(1)}_{j_3} < u^{(1)}_{j_1}$ implies $u^{(2)}_{j_1} < U_{j_1 j_3}$. □

Now we complete the proof of the $t^* = 2$ case by considering x as in (5–6). In the first case we distinguish $u^{(1)}_{j_3} \geqslant x$ when we are done by

$$m_{j_3} \leqslant x \leqslant u^{(1)}_{j_3} \leqslant u^{(1)}_{j_2} \leqslant M_{j_2}$$

and $u^{(1)}_{j_3} < x$ when we may use Lemma 2.8 by $x \leqslant M_{j_1}$ to get

$$u^{(1)}_{j_3} < x \leqslant u_{j_1 j_2} \prec U_{j_1 j_2} < U_{j_1 j_3} \leqslant M_{j_1},$$

this is in contradiction with $m_{j_1} \leqslant x \leqslant M_{j_1}$ and $m_{j_3} \leqslant u^{(1)}_{j_3} < x$ implying $U_{j_1 j_3} < x$ by the definition of $U_{j_1 j_3}$.

Finally for x as in (6) we have $m_{j_3} \leqslant u^{(1)}_{j_3} < x$ and $m_{j_1} \leqslant x$; thus $x \geqslant U_{j_1 j_3}$. We may use Lemma 2.8 by $u^{(1)}_{j_3} < x \leqslant M_{j_1}$ to obtain $U_{j_1 j_2} < U_{j_1 j_3}$. Thus we are done by $m_{j_2} \leqslant U_{j_1 j_2} < U_{j_1 j_3} \leqslant x \leqslant M_{j_1}$: chain $[m_{j_2}, M_{j_1}]$ covers x. □

As t^* increases, such a proof would become even more involved since for example in case $u^{(t^*)}_{j_2} < x \leqslant u_{j_2 j_3}$ that is easy for $t^* = 2$, for $t^* > 2$ in order to ensure a cover we must also consider $[m_{j_4}, M_{j_4}]$ since that chain also changes. As mentioned, at this point the proof could significantly be simplified by removing the choice of a tight u in Procedure ALTERNATE-PUSHDOWN since then $u^{(t^*)}_{j_2} = u_{j_2 j_3}$ and we have no such x to consider.

Since the main purpose of the next proof is to motivate the algorithm of Section 3, we cannot take the above simplifying assumption $u^{(t^*)}_{j_2} = u_{j_2 j_3}$. Instead we use an alternate proof method. Notice that Procedure REDUCE exchanges elements in the cover by traversing a virtual "augmenting path". Instead of directly proving augmenting path properties, we apply a special induction by executing the main loop of the procedure step by step and after each iteration rewinding the main algorithm. In the analogy of network flow algorithms, this may correspond to analyzing an augmenting path algorithm by choosing path edges backward from the sink, changing the flow along this edge to a preflow, and at each step proving that the remaining path augments the flow. This proof technique turns out extremely useful when proving correctness in a more complicated scenario.

The heart of the proof is a "shortest augmenting path" argument. While the sequence $u^{(t)}_j$ is decreasing for a fixed j as in (1), we can say more. Lemma 2.8 for instance states that certain pairs U_{ij} and $U_{i'j'}$ decrease in the sequence they are discovered during the course of the algorithm. In these inequalities and the more general ones that we encounter next, inequalities of this form follow from the fact that whenever $u^{(t)}_j \geqslant u^{(t)}_{j'}$, we must necessarily have $u^{(t+1)}_j \leqslant u_{jj'}$: we push down whenever there is a chance to "augment" by replacing $[m_j, M_j]$ by $[m_{j'}, M_j]$.

Proof of Theorem 2.6 *for general* t^*. We apply the following type of induction on the iterations of Procedure REDUCE. We execute the last iteration that, by Lemma 2.7,

results in a new cover $\tilde{\mathscr{I}}$ as in (2). Now we "rewind" the algorithm and rerun it with input $\tilde{\mathscr{I}}$. We complete an inductive proof by showing two facts:

- Algorithm PUSHDOWN–REDUCE produces identical $u_j^{(t)}$ for all $j \leqslant k$ and $t < t^*$ when run with $\mathscr{I}$ or $\tilde{\mathscr{I}}$,
- When run with input $\tilde{\mathscr{I}}$, the algorithm terminates in iteration $t^* - 1$.

The second fact easily follows from the first: in $\tilde{\mathscr{I}}$ no $m_{j_2} \leqslant x \leqslant u_{j_2}^{(t^*)}$ is tight since both $[m_{j_2}, M_{j_1}]$ and $[m_{j_2}, M_{j_2}]$ contains x by $x \leqslant u_{j_2}^{(t^*)} \leqslant u_{j_1}^{(t^*)}$. In $\mathscr{I}$ there is no tight element x with $u_{j_2}^{(t^*)} < x \leqslant u_{j_2 j_3}$ by the definition of Procedure PUSHDOWN; such an x cannot be tight in $\tilde{\mathscr{I}}$ since if the old chain $[m_{j_1}, M_{j_1}]$ contains x, then so is the new $[m_{j_2}, M_{j_1}]$ by $m_{j_2} \leqslant x$. Hence in $\tilde{\mathscr{I}}$ there is no tight element between m_{j_2} and $u_{j_2 j_3}$; Procedure REDUCE is called with $t^* - 1$, j_2 and j_3.

Now we prove the first fact; with this the inductive proof is complete since we have shown the base case $t^* \leqslant 2$. During the course of Algorithm PUSHDOWN–REDUCE we encounter $\tilde{m}_{j_1} = m_{j_2}$ instead of m_{j_1} when we consider values $u_{jj'}$ for j or j' equal to j_1. Since the input changes, the notion of $u_{jj'}$ is ambiguous; we use $u_{jj'} = u(m_j, m_{j'}, M_j)$ as in Definition 2.4 instead. For the sake of simplicity we continue using $U_{jj'}$; the notion is always with respect to the initial $\mathscr{I}$.

The difference in input may result in different intermediate values only in two cases: either when selecting $u(m_{j_2}, m_{j'}, M_{j_1})$ instead of $u(m_{j_1}, m_{j'}, M_{j_1})$ for defining $u_{j_1}^{(t)}$ or when selecting $u(m_j, m_{j_2}, M_j)$ instead of $u(m_j, m_{j_1}, M_j)$ for defining $u_j^{(t)}$. Thus we are done by showing $u(m_{j_2}, m_{j'}, M_{j_1}) = (m_{j_1}, m_{j'}, M_{j_1})$ and $u(m_j, m_{j_2}, M_j) = u(m_j, m_{j_1}, M_j)$; we prove it via the next lemma. The actual proof details hide the simple reasoning that changing chain $[m_{j_1}, M_{j_1}]$ to chain $[m_{j_2}, M_{j_1}]$ may not have effects on the algorithm above $U_{j_1 j_2}$.

Lemma 2.9. *If $U_{j_1 j_2} < U_{j_1 j'}$, then $u(m_{j_2}, m_{j'}, M_{j_1}) = u(m_{j_1}, m_{j'}, M_{j_1})$. And if $U_{j_1 j_2} < U_{j j_1}$, then $u(m_j, m_{j_2}, M_j) = u(m_j, m_{j_1}, M_j)$.*

Proof. The first part follows if $u = u(m_{j_1}, m_{j'}, M_{j_1}) \geqslant m_{j_2}$: elements $U = U(m_{j_1}, m_{j'}, M_{j_1})$ and u both belong to chain $[m_{j_2}, M_{j_1}]$ and they satisfy $u \prec U$ by Definition 2.4. Since we also have $u \not\geqslant m_{j'}$ and $U \geqslant m_{j'}$, elements u and U are minimum and maximum in chain $[m_{j_2}, M_{j_1}]$ as required. And to show $u \geqslant m_{j_2}$ we notice $m_{j_1} \leqslant u \prec U$ and $m_{j_1} \leqslant U_{j_1 j_2} < U_{j_1 j'} = U$. This means both u and $U_{j_1 j_2}$ belong to the chain between m_{j_1} and U, thus they are comparable by the unique path property. And since $u \prec U$, we must also have $U_{j_1 j_2} \leqslant u$. This implies $m_{j_2} \leqslant U_{j_1 j_2} \leqslant u$, as required.

For the second part we have to show $u = u(m_j, m_{j_1}, M_j) \not\geqslant m_{j_2}$ and $U = U_{j j_1} = U(m_j, m_{j_1}, M_j) \geqslant m_{j_2}$. The second inequality follows by $m_{j_2} \leqslant U_{j_1 j_2} < U_{j j_1} = U$. We prove the first inequality by assuming $m_{j_2} \leqslant u$ by contradiction. By Definition 2.4 we have $m_{j_2} \leqslant u \prec U$ then; together with the above inequality $m_{j_2} \leqslant U$ we get that both u and $U_{j_1 j_2}$ are between m_{j_2} and U, and thus they are comparable by the unique path property. And since $u \prec U$, we must also have $U_{j_1 j_2} \leqslant u$. By $m_{j_1} \leqslant U_{j_1 j_2} \leqslant u$ we reached a contradiction with the definition of $u = u(m_j, m_{j_1}, M_j)$. □

We proceed by proving the inequalities of the lemma for all the steps of the algorithm that involve the new value $\tilde{m}_{j_1} = m_{j_2}$. First we consider the value of $u_{j_1}^{(t)}$; for $t = 1$ it is determined in Procedure ALTERNATE-INIT and for the general t in Procedure PUSHDOWN. The former procedure starts iterations with M_{j_1} and the latter with $u_{j_1}^{(t-1)}$. With respect to inputs $\mathscr{I}$ and $\tilde{\mathscr{I}}$, values of M_{j_1} are identical. We may also assume the $u_{j_1}^{(t-1)}$ remain identical by applying induction on t. Thus $u_{j_1}^{(t)}$ is the final element of a decreasing sequence of values $u_{j_1 j'} \geqslant u_{j_1}^{(t)}$; here $u_{j_1 j'} = u(m_{j_1}, m_{j'}, M_{j_1})$ for input $\mathscr{I}$ and $\tilde{u}_{j_1 j'} = u(m_{j_2}, m_{j'}, M_{j_1})$ for input $\tilde{\mathscr{I}}$. We prove these two poset elements are identical by combining the next lemma with Lemma 2.9.

In the next three lemmas we use u_{ji} and U_{ji} all with respect to the original input $\mathscr{I}$.

Lemma 2.10. *Let $u_{j_1}^{(t^*)} \geqslant u_{j_2}^{(t^*)}$ and for some $j \neq j_1, j_2$ and $t \leqslant t^*$ let $u_{j_1}^{(t)} \leqslant u_{j_1 j}$. Then $U_{j_1 j_2} < U_{j_1 j}$.*

Proof. We are done by

$$U_{j_1 j_2} \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t)} \leqslant u_{j_1 j} \prec U_{j_1 j},$$

where the first inequality follows by $m_{j_2} \leqslant u_{j_2}^{(t^*)} \leqslant u_{j_1}^{(t^*)}$ and the second by (1) and $t^* \geqslant t$. □

Next we turn to $u_j^{(t)}$ for $j \neq j_1$; as before we consider Procedures ALTERNATE-INIT and PUSHDOWN for $t = 1$ and $t > 1$, respectively. Again we use induction for t. The procedures may select different values $u(m_j, m_{j_1}, M_j)$ and $u(m_j, m_{j_2}, M_j)$ for inputs $\mathscr{I}$ and $\mathscr{I}$. We prove these elements are identical by combining Lemma 2.9 and the two complementary Lemmas 2.12 and 2.13, both using the next simple fact.

Lemma 2.11. *For some $j \neq j_1, j_2$ and $t < t^*$ assume that $u_{j_1}^{(t^*)} \geqslant U_{j_1 j_2}$ and $U_{j j_1}$ exist and that the value $u(m_j, m_{j_1}, M_j)$ is used when defining $u_j^{(t)}$ in Procedure* PUSHDOWN *or* ALTERNATE-INIT. *Then $U_{j_1 j_2}$ and $U_{j j_1}$ are comparable.*

Proof. First we show $u_{j_1}^{(t)} \leqslant M_j$. By the call condition $M_j = M_{j_1}$ of Procedure ALTERNATE-INIT this is immediate for $t = 1$. For $t > 1$ this follows since when j_1 appears in Procedure PUSHDOWN, then the condition $u_{j_1}^{(t-1)} \leqslant u$ is satisfied with u as the running value in Procedure PUSHDOWN satisfying $u \leqslant u_j^{(t-1)}$. Hence we get

$$u_{j_1}^{(t)} \leqslant u_{j_1}^{(t-1)} \leqslant u \leqslant u_j^{(t-1)} \leqslant M_j.$$

The claim follows since both $U_{j_1 j_2}$ and $U_{j j_1}$ are not less than m_{j_1} and $U_{j j_1} \leqslant M_j$ by definition; finally by using the above sequence of inequalities $U_{j_1 j_2} \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t)} \leqslant M_j$. □

Lemma 2.12. *Assume that under the condition of Lemma* 2.11 *we have that $U_{j_1 j}$ exists. Then $U_{j j_1} > U_{j_1 j_2}$.*

Proof. Since U_{jj_1} and $U_{j_1j_2}$ are comparable by Lemma 2.11, we are done by deriving a contradiction from $U_{jj_1} \leqslant U_{j_1j_2}$. We get

$$u_j^{(t)} \leqslant u_{jj_1} \prec U_{jj_1} \leqslant U_{j_1j_2} \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t)};$$

contradiction follows since by definition Procedure PUSHDOWN in iteration $t+1$ applies for j_1 and j to yield $U_{jj_1} \succ u_{j_1j} \geqslant u_{j_1}^{(t+1)} \geqslant u_{j_1}^{(t^*)}$. Here the second inequality follows by Procedure PUSHDOWN and the last by (1) and $t^* > t$. □

Lemma 2.13. *Assume that under the condition of Lemma* 2.11 *we have that* U_{j_1j} *does not exist. Then* $U_{jj_1} > U_{j_1j_2}$.

Proof. As in Lemma 2.12 we are done by deriving a contradiction from $U_{jj_1} \leqslant U_{j_1j_2}$: inequality $m_{j_1} \leqslant U_{jj_1} \leqslant U_{j_1j_2} \leqslant M_{j_1}$ together with $m_j \leqslant U_{jj_1}$ imply the existence of U_{j_1j}. □

3. Interval covers and vertex connectivity augmentation

In this section we describe the main algorithm of the paper that solves certain directed edge augmentation problems via a reduction to a poset covering problem. As shown in Section 3.1, this covering problem is a slight generalization of that considered by Frank and Jordán [10]. Thus our algorithm applies among others to the task of increasing directed vertex connectivity or directed S–T edge connectivity by one, and even of finding the minimum number of rectangles covering a horizontally convex rectilinear area.

Frank and Jordán [10] give an algorithm for a more general class of problems where, in our terminology, poset elements are weighted by a non-negative supermodular function. However their algorithm is based on the ellipsoid method [15]. We give a combinatorial algorithm (for the unweighted problem) in Sections 3.2 and 3.3. The algorithm uses an approach completely different from that of the recent result of Frank [9].

3.1. Poset properties of the Frank–Jordán set pairs

Frank and Jordán [10] introduce systems of set pairs closed under a certain "skew intersection operation defined next. Let two members (S,T) and (S',T') be called *dependent* if both $S \cap S'$ and $T \cap T'$ are nonempty; otherwise they are *independent*. Then for all dependent pairs,

$$(S \cap S', T \cup T'),\ (S \cup S', T \cap T')$$

are also members of the set system. They prove the following theorem:

Theorem 3.1 (Frank and Jordán [10]). *The minimum number of edges* $e=(v_1,v_2)$ *such that for all* (S,T) *there exists an edge with* $v_1 \in S$, $v_2 \in T$ *is equal to the maximum number of pairwise independent elements in the system of set pairs.*

We give an alternate proof of a slight strengthening of this theorem stated as a poset covering problem. The proof is via a combinatorial algorithm.

Definition 3.2. Consider a poset $\mathscr{P}$; let $u, v \in \mathscr{P}$ be called *dependent* if $\exists\, m, M$ with $m \leqslant u \leqslant M$ and $m \leqslant v \leqslant M$; otherwise they are *independent*. We say that a poset $\mathscr{P}$ satisfies the *strong interval property* if for all dependent u and $v \in \mathscr{P}$ two commutative, associative and distributive operations $\vee$ and $\wedge$ are uniquely defined as

$$s \vee t = \min\{x\colon x \geqslant s, x \geqslant t\}; s \wedge t = \max\{x\colon x \leqslant s, x \leqslant t\}; \tag{7}$$

furthermore for all minimal and maximal poset elements $m \leqslant M$,

$$m \leqslant u \wedge v \leqslant M \text{ implies } m \leqslant u \leqslant M \textbf{ or } m \leqslant v \leqslant M, \tag{8}$$

finally the same holds with $u \wedge v$ replaced by $u \vee v$. We say $\{x\colon m \leqslant x \leqslant M\}$ is the *interval* $[m, M]$; a collection of intervals $\mathscr{I} = \{[m_i, M_i] : i \leqslant k\}$ *covers* the poset if

$$\forall x \in \mathscr{P}\ \exists\, i : m_i \leqslant x \leqslant M_i.$$

An element v with $v \in [m_i, M_i]$ for a unique i in a cover is called *tight*.

We may rephrase the strong interval property in the terms of intervals:

$$u \wedge v \in [m, M] \textbf{ or } u \vee v \in [m, M] \text{ implies } u \in [m, M] \textbf{ or } v \in [m, M]. \tag{9}$$

The property implies that

$$\text{if } u \text{ and } v \text{ are tight, then so are } \textit{both } u \wedge v \textit{ and } u \vee v : \tag{10}$$

to prove, assume w.l.o.g. that $u \vee v$ is not tight, i.e. $u \vee v \in [m, M] \cap [m', M']$ for two distinct intervals. Since $u \wedge v$ is covered, it is contained by at least one interval $[m'', M'']$. If this interval is identical to either of the previous two, then $u, v \in [m'', M'']$ and contradiction follows since another interval contains u or v by (9). And if all three intervals are distinct, then again each contains u or v by (9) and thus one of u and v is not tight.

Theorem 3.3. *Let $\mathscr{P} \subseteq \{(S, T) : S \subseteq V_1, T \subseteq V_2\}$ such that for all dependent $s = (S, T)$ and $t = (S', T')$,*

$$s \wedge t = (S \cap S', T \cup T'),\ s \vee t = (S \cup S', T \cap T') \in \mathscr{P}.$$

For any $s = (S, T)$ and $t = (S', T')$ let $s \leqslant t$ iff $S \subseteq S'$ and $T \supseteq T'$. Then operations $\vee$, $\wedge$ and $\leqslant$ over $\mathscr{P}$ satisfy the strong interval property. Furthermore subfamilies

$$\{(S, T) : v_1 \in S, v_2 \in T\}$$

for pairs $v_1 \in V_1$, $v_2 \in V_2$ are precisely the maximal intervals of $\mathscr{P}$.

Proof. Property (7) follows directly by the properties of set union, intersection and containment. To show the equivalence of maximal intervals and subfamilies covered by edges, consider an edge (v_1, v_2) first. Since all set pairs covered by the edge are dependent, we may take intersection and unions to find unique minimal and maximal

pairs; the maximal interval consisting of all pairs between the two are covered by (v_1, v_2). And given a maximal interval $[m, M] = [(S, T), (S', T')]$, we may take an edge (v_1, v_2) with $v_1 \in S$ and $v_2 \in T'$. This edge covers the entire interval. Assume it also covers some v with $v \not\geqslant m$ or $v \not\leqslant M$. In the first case $v \wedge m < m$ and in the second case $v \vee M > M$ exists, contradicting the extremity of m or M.

For (8) take an edge (v_1, v_2) covering the interval $[m, M]$. It suffices to show this edge covers either $u = (S, T)$ or $v = (S', T')$. Notice $v_1 \in S \cap S'$ and $v_2 \in T \cup T'$. The former implies $v_1 \in S$ and $v_1 \in S'$ while the latter implies $v_2 \in T$ or $v_2 \in T'$. The claim follows. □

Before giving our algorithm, we state our main result as a min–max formula for posets with the strong interval property. By Theorem 3.3 our result (slightly) extends Theorem 3.1.

Theorem 3.4. *For a poset $\mathscr{P}$ satisfying the strong interval property, the minimum number of intervals covering $\mathscr{P}$ is equal to the maximum number of pairwise independent elements of $\mathscr{P}$.*

3.2. The algorithm

The main difference of the interval cover problem compared to the chain cover problem is that the dependence of u and u' does not imply comparability; we may also have a pair $[m, M]$ for an incomparable u and u' with $m \leqslant u \leqslant M$ and $m \leqslant u' \leqslant M$. In both (dependent) cases $u \vee u'$ and $u \wedge u'$ exist. While the main structure of Algorithm PUSHDOWN–REDUCE remains unchanged compared to Section 2, the two procedures will be different and restated under the same name in this section. The name conflict will cause no confusion since we no longer use the procedures of the previous section.

As for the algorithmic issues, we have to be more careful in an interval cover algorithm than in a Dilworth algorithm, since in the latter the input size is the poset size while in the former we typically have an exponential size implicitly given poset such as a set of (directed) min-cuts. The steps of the algorithm are easily checked to be polynomial in the number total possible different intervals and the length of a longest chain in the poset. For example for graph edges and cuts, the former is $O(n^2)$ while the latter is $O(n)$. We give no analysis in more detail; the efficiency of the algorithms depend on oracle implementations that is beyond the scope of this paper.

The algorithm takes $\vee$ and $\wedge$ of poset elements; compares poset elements (follows from taking $\vee$ and $\wedge$, provided we are able to tell identity of poset elements); and computes the minimum and maximum tight elements of intervals. While the first two steps are trivial, we may, in connectivity augmentation applications, implement the third one as a single min-cut computation.

We begin the description of Procedure PUSHDOWN by giving a rule that, for the input u_j, selects non-independent elements u_i for further processing. As a main difference compared to the Dilworth algorithm, the algorithm may not necessarily consider i with $u_i \geqslant u_j$ since possibly all non-independent pairs could be incomparable with only $u_i \wedge u_j$ existing. The lemma below gives a sufficient rule of choice so that whenever we have

dependent $\{u_j^{(t)}\}$, we may progress with a PUSHDOWN step in Algorithm PUSHDOWN–REDUCE.

Lemma 3.5. *If the set* $\{u_i^{(t)}: i \leqslant k\}$ *of tight poset elements with* $m_i \leqslant u_i \leqslant M_i$ *for* $i \leqslant k$ *is not pairwise independent, then there exists* $i \neq j$ *with* $u_i \wedge u_j \geqslant m_j$ *and* $u_i \vee u_j \leqslant M_i$.

Proof. The claim is immediate if $u_i \geqslant u_j$; for $u_j \geqslant u_i$ we exchange the role of i and j. Finally if u_i and u_j are incomparable, then consider j^+ and j^- with

$$m_{j^+} \leqslant u_i \vee u_j \leqslant M_{j^+} \quad \text{and} \quad m_{j^-} \leqslant u_i \wedge u_j \leqslant M_{j^-}.$$

Since both u_i and u_j are tight, we must have j^+ and j^- equal to i or j by (8). By possibly exchanging the role of i and j, we may assume $j^+ = i$ and $j^- = j$ or $j^- = i$. In the first case we are done; in the second case $m_i \leqslant u_j \leqslant M_i$ contradicts that u_j is tight. □

The Dilworth algorithm of Section 2 is based on poset elements

$$u(m, m', M) = \max\{x: m \leqslant x \leqslant M \text{ and } x \not\geqslant m'\},$$

$$U(m, m', M) = \min\{x: m \leqslant x \leqslant M \text{ and } x \geqslant m'\}.$$

These elements are well defined in posets with the strong interval property as well: if two elements x and x' satisfy the maximization (minimization) requirement, then so is their union and intersection: they both belong to $[m, M]$ by (7); if $m' \leqslant U_1$ and $m' \leqslant U_2$, then $m' \leqslant U_1 \wedge U_2$ again by (7); finally if $m' \not\leqslant U_1$ and $m' \not\leqslant U_2$, then we cannot have $m' \leqslant U_1 \wedge U_2$ by (8). We use the short notation

$$u_{ji} = u(m_j, m_i, M_j), \quad U_{ji} = U(m_j, m_i, M_j)$$

whenever it causes no ambiguity.

A dramatic difference compared to posets with the unique path property is that for a fixed j there is no unique minimal $\{u_{ji} : u_i \leq u_j\}$ as in Procedure PUSHDOWN of Section 2. Instead we will have to consider a set of possible j' with incomparable $u_{jj'}$; however, we may take arbitrary intersection $\wedge$ since $m_j \leqslant u_{ji} \leqslant M_j$ for all i. By taking intersections we preserve the decreasing sequence of $u_j^{(t)}$ as in (1).

We are ready to state Procedure PUSHDOWN. We consider all i as in the lemma and take the intersection v of all the $u_{ji} \wedge u_j^{(t)}$ first. Then we either let $u_j^{(t+1)}$ be the maximum tight element below v, or if that does not exist, we may start reducing the cover with interval j.

We remark that $[m_i, M_i]$, $u_i^{(t)}$ and the value of t are global in Algorithm PUSHDOWN–REDUCE, hence in particular the procedure below has access to the intervals.

We also mention that in Section 3.4 we give an alternate algorithm that, instead of computing minimum and maximum tight elements, uses u_{ji} and U_{ji}.

The reason why poset elements u_{ji} and U_{ji} behave in a more complex way is that we no longer have $u_{ji} \prec U_{ji}$. Instead in what follows we will use the following intersection property of the u_{ji} as a substitute for the predecessor of U_{ji}.

Procedure PUSHDOWN(j, $\{u_i\}$)
  $v \leftarrow \bigwedge\{u_{jj'} : m_{j'} \leqslant u_j$ and $u_{j'} \leqslant M_j$ for $m_{j'} \neq m_j\} \wedge u_j$
  if there is no tight x with $m_j \leqslant x \leqslant v$ **then**
    exit with reduced cover REDUCE(j)
  return the maximum tight element x with $m_j \leqslant x \leqslant v$

Lemma 3.6. *For $m \leqslant x \leqslant M$ and $m' \leqslant M$, element $u = u(m,m',M) \wedge x$ is maximal with $m \leqslant u \leqslant x$ and $u \not\geqslant U(m,m',M)$.*

Proof. By definition $u \leqslant u(m,m',M)$ implies $u \not\geqslant U(m,m',M)$. Let $m \leqslant u' \leqslant x$ and $u' \not\geqslant U(m,m',M)$ for some $u' \not\leqslant u$. Notice $u \vee u' \not\geqslant U(m,m',M)$ since otherwise we would have $m' \leqslant u \vee u' \leqslant M$ implying $m' \leqslant u \leqslant M$ or $m' \leqslant u' \leqslant M$ by the strong interval property (8) and contradicting $u \not\geqslant U(m,m',M)$ and $u' \not\geqslant U(m,m',M)$.

Since $m \leqslant u \vee u' \leqslant x$ and $u \vee u' \not\geqslant U(m,m',M)$, we may assume $u' > u$. However $u < u' \leqslant x$ implies $u' \not\leqslant u(m,m',M)$, since otherwise $x \wedge u(m,m',M) \geqslant u' > u$ would hold. We are done since $u' \not\leqslant u(m,m',M)$ implies $u' \geqslant U(m,m',M)$ by the maximality of $u(m,m',M)$ and the minimality of $U(m,m',M)$. □

3.3. Procedure REDUCE for interval covers

When describing Procedure REDUCE for interval covers, we face the difficulty that typically there is no unique j' with $u_{jj'}$ minimum for $m_{j'} \leqslant u_j^{(t)}$ and $u_{j'}^{(t)} \leqslant M_j$. In Section 2 we had a unique minimizer j' and defined a sequence j_s such that $u_{j_s j_{s-1}}$ is minimum in the above sense. Instead our rule for choosing j_s is based on the next lemma using the fact that we take the intersection of all $u_{jj'}$. The lemma has to be applied with $u = u_j$, $[m,M] = [m_j, M_j]$ and $[m',M'] = [m_{j'}, M_j]$ for j' with $u_{j'} \leqslant M_j$.

Lemma 3.7. *For some $m \leqslant u \leqslant M$ let $v = \bigwedge\{u(m,m',M) : u \in [m',M']$ for certain $[m',M']$ with $m' \neq m\} \wedge u$. For all $x \leqslant u$ and $x \not\leqslant v$ there exists an $[m',M']$ in the definition of v such that $x \in [m',M']$.*

Proof. Since $x \not\leqslant v$, there exists $[m',M']$ with $x \not\leqslant u \wedge u(m,m',M)$. By Lemma 3.6 $y = u \wedge u(m,m',M)$ is maximum with $y \leqslant u$ and $y \not\geqslant m'$. Since $x \not\leqslant y$, we get $y < x \vee y$ whence $m' \leqslant x \vee y$ follows by the maximality of y. However, then $x \vee y$ is contained by $[m',M']$ and so is x by the strong interval property (9). The proof is complete. □

Now we give the procedure. For a pair of corresponding values of s and t in Algorithm PUSHDOWN–REDUCE, let

$$u = \min\{m_{j_s} \leqslant u \leqslant u_{j_s}^{(t)} : u \text{ is tight}\}.$$

The procedure simply chooses a j_{s+1} with $m_{j_{s+1}} \leqslant u$ and changes interval $[m_{j_s}, M_{j_s}]$ to $[m_{j_{s+1}}, M_{j_s}]$. The new interval also contains the minimum tight element and thus maintains the cover.

For the correctness of the procedure we notice that u exists, since $u_{j_s}^{(t)}$ is tight and thus we minimize over a non-empty set. The minimum is unique, since for two tight elements u_1 and u_2 both satisfying the minimization requirements, $u = u_1 \wedge u_2$ also satisfies the requirements with the exception that it is not necessarily tight. However, in that case $m_j \leqslant u \leqslant M_j$ for some $j \neq j_s$ as well; by the strong interval property either u_1 or u_2 is not tight then.

Procedure REDUCE(j)
  $j_1 \leftarrow j$;
  for $t = t^*, \ldots, 1$; $s = 1, \ldots, t^*$ **do**
    $u \leftarrow \min\{m_{j_s} \leqslant u \leqslant u_{j_s}^{(t)} : u \text{ is tight}\}$
    $j_{s+1} \leftarrow$value of $j \neq j_s$ with $m_j \leqslant u$ and $u_j^{(t)} \vee u_{j_s}^{(t)} \leqslant M_{j_s}$
    $m_{j_s} \leftarrow m_{j_{s+1}}$
  return $\{[m_i, M_i] : i \leqslant k,\ i \neq j_{t^*+1}\}$

Finally, we show j_{s+1} exists; for this we apply Lemma 3.7 for the minimum tight u and element v as in Procedure PUSHDOWN. The lemma immediately applies if $u \not\leqslant v$. We complete the description by showing $u \not\leqslant v$. This is immediate for $s = 1$ and $t = t^*$ since Procedure PUSHDOWN calls Procedure REDUCE only if there are no tight elements below v.

For all other $s > 1$ we want to use the above argument that there is no tight element between v and m_{j_s} when Procedure PUSHDOWN is called in iteration t of the main algorithm. While this argument clearly fails for the initial cover since $u_{j_s}^{(t+1)}$ is tight, this element and all elements below it become no longer tight when $m_{j_{s-1}}$ is replaced by $\tilde{m}_{j_{s-1}} = m_{j_s}$ as shown in the next lemma. Notice, however, that other intervals $[m_{j_i}, M_{j_i}]$ for $i < s$ may have changed that affect the tightness of elements between v and $u_{j_s}^{(t+1)}$ (initially none of them are tight by Procedure PUSHDOWN). The proof for $s > 1$ is postponed to the next section where it will be proved by the rewinding technique of Section 2.3, i.e. rerunning the algorithm with a new interval cover arising by executing a single iteration of Procedure REDUCE. There we show that the properties of the $u_j^{(t)}$ are preserved throughout the steps of Procedure REDUCE, completing the correctness proof of Procedure REDUCE.

Lemma 3.8. *As in Procedure* REDUCE *let* $v = \bigwedge\{u_{j_s j'} : m_{j'} \leqslant u_{j_s}^{(t)}$ *and* $u_{j'}^{(t)} \leqslant M_{j_s}$ *for* $m_{j'} \neq m_{j_s}\} \wedge u_{j_s}^{(t)}$. *With respect to the interval system of iteration* s *of Procedure* REDUCE, *if there is no tight* $u_{j_s}^{(t+1)} \leqslant u \leqslant v$, *then there is no tight* $m_{j_s} \leqslant u \leqslant v$.

Proof. Let us assume $m_{j_s} \leqslant u \leqslant v$ is tight; notice $u \not\geqslant u_{j_s}^{(t+1)}$. First we show u is contained in both $[m_{j_s}, M_{j_s}]$ and $[m_{j_s}, M_{j_{s-1}}]$ of the current interval system in iteration s; this follows by

$$m_{j_s} \leqslant u \leqslant v \leqslant u_{j_s}^{(t)} \leqslant u_{j_s}^{(t)} \vee u_{j_{s-1}}^{(t)} \leqslant M_{j_{s-1}}, M_{j_s}.$$

Let us define $u' = u_{j_s}^{(t+1)} \vee u$; since u is tight by assumption and $u_{j_s}^{(t+1)}$ by definition, we get by the strong interval property (10) that u' is tight. However $u_{j_s}^{(t+1)} < u' = u_{j_s}^{(t+1)} \vee u \leqslant v$, hence we find a tight element between $u_{j_s}^{(t+1)}$ and v. □

3.4. ALTERNATE-PUSHDOWN: *a different algorithm*

The current Procedure PUSHDOWN uses a mix of steps requiring values u_{ji} and maximum tight elements in a form most convenient for describing Procedure REDUCE. While for connectivity augmentation it is advantageous to base the algorithm on selecting maximum tight elements, for certain problems this might require oracles that are hard to implement. We give alternate implementations based purely either on the values of u_{ji} as in the original Procedure PUSHDOWN in Section 2 or on selecting maximum tight elements. Notice that the current algorithm resembles Procedures ALTERNATE-INIT and ALTERNATE-PUSHDOWN of the Dilworth algorithm.

The key step in switching between the selection of tight elements of $[m, M]$ below certain v and intersecting certain u_{ji} is Procedure PUSH-TO-TIGHT described as follows. We have nothing to do if v is tight. Otherwise there is a subset of $\mathscr{I}$ with $m' \leqslant v \leqslant M'$ for $[m', M'] \in \mathscr{I}$. We take $v \leftarrow \bigwedge_i u(m, m', M) \wedge v$ for all these $[m', M']$; the new v is guaranteed not to be contained in any of these intervals. And if v is still not tight, we simply iterate the procedure with the new value of v.

Procedure PUSH-TO-TIGHT($v_1, [m, M], \mathscr{I}$)
  $s \leftarrow 1$
  $\mathscr{I} \leftarrow \mathscr{I} - [m, M]$
  while v_s is not tight **do**
    if $\exists [m', M'] \in \mathscr{I}$ with $m' = m \leqslant v_s \leqslant M'$ **then**
      return NULL
    else
      $\mathscr{I}' \leftarrow \{[m', M'] \in \mathscr{I} : m' \leqslant v_s \leqslant M' \text{ for } m' \neq m\}$
      $\mathscr{I} \leftarrow \mathscr{I} - \mathscr{I}'$
      $v_{s+1} \leftarrow \bigwedge \{u(m, m', M) : m' \in \mathscr{I}'\} \wedge v_s$
      $s \leftarrow s + 1$
  return v_s

We summarize the modified algorithm now. Instead of taking $u_j^{(1)}$ as the maximum tight element, we take PUSH-TO-TIGHT(M_j). And instead of taking $u_j^{(t)}$ as the maximum tight element below certain v, we again call PUSH-TO-TIGHT(v).

Tight elements are finally chosen in Procedure REDUCE to select j_{s+1} such that $u \not\geqslant u_{j_s j_{s+1}}$ for the minimum tight $u \leqslant u_{j_s}^{(t)}$; equivalently this means that there is no tight element below $u_{j_s}^{(t)}$ with respect to $\mathscr{I} \cup \{(m_{j_{s+1}}, M_{j_s})\}$. We may check this condition by running Procedure PUSH-TO-TIGHT(u) for these sets of intervals for all possible j_{s+1}.

We prove correctness by showing we get the same values $u_j^{(t)}$ in the modified algorithm. This is immediate from the following theorem.

Lemma 3.9. *Element* v_s=PUSH-TO-TIGHT$(v_1,[m,M],\mathscr{I})$ *is the maximum element* $m \leqslant v_s \leqslant v_1$ *not contained by any* $[m',M'] \in \mathscr{I} - [m,M]$.

Proof. First notice the removal of $\mathscr{I}'$ from $\mathscr{I}$ has no effect on the values of v_s since as soon as $v_s \leqslant u(m,m',M)$, we may no longer have $[m',M'] \in \mathscr{I}'$. Now by contradiction let $v \neq v_s$ be maximum tight; $v \not\leqslant v_s$. Since $v \leq v_1$, we may let $t < s$ be minimum with $v \leqslant v_t$. By Lemma 3.7 there exists $[m',M']$ in the definition of v_{t+1} such that $m' \leqslant v$. We have reached contradiction by $m' \leqslant v \leqslant v_t \leqslant M'$. □

Now we give an opposite substitution in Procedure PUSHDOWN, replacing u_{ji} by the selection of maximum tight elements.

Lemma 3.10. *Poset element* v *as in Procedure* PUSHDOWN *is the maximum element* $m_j \leqslant x \leqslant u_j$ *not contained by any* $[m_{j'},M_j]$ *with* $m_{j'} \leqslant u_j$ *and* $u_{j'} \leqslant M_j$.

Proof. Consider Procedure PUSH-TO-TIGHT with $\mathscr{I} = \{[m_{j'},M_j]$ with $m_{j'} \leqslant u_j$ and $u_{j'} \leqslant M_j\}$. The procedure terminates in a single execution of the main loop since $\mathscr{I}' = \mathscr{I}$ in the first step. Since v_2 is the same as v in Procedure PUSHDOWN, the claim follows by Lemma 3.9. □

Finally, we mention that we may at any time abort the execution of Procedure PUSH-TO-TIGHT and proceed by selecting a maximum tight element below the current v_s. This fact turns out useful in the proof of the main theorem.

Lemma 3.11. *Consider any iteration* s of *Procedure* PUSH-TO-TIGHT; *let* $\mathscr{I}''$ *be an arbitrary subset of* $\mathscr{I}'$. *Let us define* v'_s *by considering* $\mathscr{I}''$ *instead of* $\mathscr{I}'$. *Then the output of Procedure* PUSH-TO-TIGHT *is equal to the maximum* $m \leqslant x \leqslant v'_s$ *not contained by any* $[m',M'] \in \mathscr{I} - \mathscr{I}''$.

Proof. Apply Lemma 3.9 with $\mathscr{I} \leftarrow \mathscr{I} - \mathscr{I}''$ and $v_1 = v'_s$. □

3.5. Proofs

We prove by the rewinding technique as in Theorem 2.6. The first step is a single elementary interval endpoint change that yields a new cover $\tilde{\mathscr{I}}$ similar to that of the previous section. We state and prove the main theorem afterward.

Lemma 3.12. *Let* t^* *and* j_t *be as in Procedure* REDUCE. *Then the set* $\tilde{\mathscr{I}}$ *arising by replacing* $[m_{j_1},M_{j_1}]$ *by* $[m_{j_2},M_{j_1}]$ *in* $\mathscr{I}$ *is also an* (*interval*) *cover.*

Proof. To show that the new $\tilde{\mathscr{I}}$ is indeed a cover, we have to consider $x \in \mathscr{P}$ with $m_{j_1} \leqslant x \leqslant M_{j_1}$. Let $u = \min\{m_{j_1} \leqslant u \leqslant u_{j_1}^{(t)} : u \text{ is tight}\}$ be as in Procedure REDUCE. We split the set of such x into $u \leqslant x$ and $u \not\leqslant x$ corresponding to (3–4). The first case is immediate by $m_{j_2} \leqslant u \leqslant x \leqslant M_{j_1}$. In the second case $x \wedge u < u$; this element is not

tight since u is minimum tight by definition. The strong interval property (10) implies that x is not tight then. □

Theorem 3.13. *If Procedure* REDUCE *is called by Algorithm* PUSHDOWN–REDUCE, *then the output of the procedure is an interval cover of the poset.*

Proof. Let us begin with the $t^* = 1$ case. We have to show all elements x with either $m_{j_1} \leqslant x \leqslant M_{j_1}$ or $m_{j_2} \leqslant x \leqslant M_{j_2}$ are covered after $[m_{j_1}, M_{j_1}]$ and $[m_{j_1}, M_{j_1}]$ are replaced by $[m_{j_2}, M_{j_1}]$. For $m_{j_1} \leqslant x \leqslant M_{j_1}$ this follows by Lemma 3.12. Now consider a tight $m_{j_2} \leqslant x \leqslant u_{j_2}^{(1)}$; we show $x \leqslant M_{j_1}$. By definition $u_{j_2}^{(1)}$ is the maximal tight element of the interval $[m_{j_2}, M_{j_2}]$, thus $x \leqslant u_{j_2}^{(1)}$. However by $t^* = 1$ and the definition of j_2, $m_{j_2} \leqslant x \leqslant u_{j_2}^{(1)} \leqslant M_{j_1}$, as required.

For general t^* we prove the correctness of Algorithm PUSHDOWN–REDUCE with the rewinding technique of Section 2. We are done if we show:

- inductively for all $t = 1, \ldots, t^* - 1$ that $u_j^{(t)}$ is identical when the algorithm is run with $\mathscr{I}$ or the modified $\tilde{\mathscr{I}}$ as in Lemma 3.12; and
- the algorithm, when run with $\tilde{\mathscr{I}}$, terminates with value $t = t^* - 1$.

To prove the second claim, we assume the first claim and let

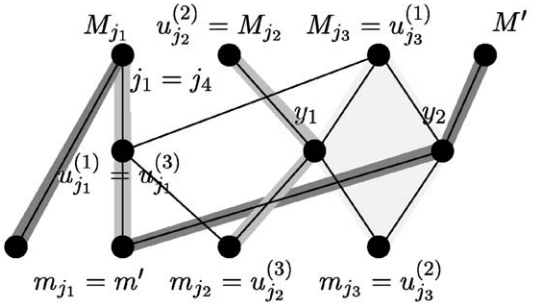
$$v = \bigwedge \{u_{j_2 j'} : m_{j'} \leqslant u_{j_2}^{(t^*-1)} \text{ and } u_{j'} \leqslant M_{j_2} \text{ for } m_{j'} \neq m_{j_2}\} \wedge u_{j_2}^{(t^*-1)}$$

as in Procedure PUSHDOWN$(j_2, \{u_i^{(t^*-1)}\})$. We have to show there is no tight $m_{j_2} \leqslant u \leqslant v$. By Lemma 3.8, with respect to $\tilde{\mathscr{I}}$, the interval system of iteration $s = 2$ of Procedure REDUCE, such a u must satisfy $u_{j_2}^{(t^*)} \leqslant u \leqslant v$. By the definition of $u_{j_2}^{(t^*)}$ in Procedure PUSHDOWN, u is not tight with respect to the initial $\mathscr{I}$. Thus in $\tilde{\mathscr{I}}$ we had to remove one chain containing u; this means $m_{j_1} \leqslant u \leqslant M_{j_1}$. However, then we also added a new chain $[m_{j_2}, M_{j_1}]$ that contains u. Hence u cannot be tight, and by running the algorithm with input $\tilde{\mathscr{I}}$, Procedure PUSHDOWN$(j_2, \{u_i^{(t^*-1)}\})$ exits to Procedure REDUCE.

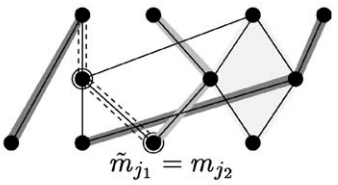
We complete the proof by showing the first claim above. We have to investigate the process of defining $u_j^{(t)}$ in Algorithm PUSHDOWN–REDUCE. By Lemma 3.9 we saw the definition may use values $u(m_j, m_{j'}, M_j)$ or, by Lemma 3.11, a combination of these values and the selection of maximum tight elements. We distinguish two cases:

(1) If $j_1 = j$, then $u(m_{j_1}, m_{j'}, M_{j_1})$ changes to $u(m_{j_2}, m_{j'}, M_{j_1})$ for all j'. We show these poset elements are identical in an argument virtually identical to Lemmas 2.9 and 2.10: we prove $U_{j_1 j_2} \leqslant u_{j_1 j}$; thus the change in $[m_{j_1}, M_{j_1}]$ below $U_{j_1 j_2}$ does not affect $u(m_{j_1}, m_{j'}, M_{j_1})$.

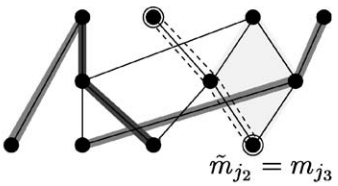
(2) If $j_1 = j'$, then for this single j' we have $u(m_j, M_{j_1}, M_j)$ changing to $u(m_j, M_{j_2}, M_j)$. Unlike in the previous case or in the algorithm of Section 2 we no longer have the equality of these elements, as shown in the example of Fig. 2. Instead as in Lemma 3.11 we abort the process of selecting $u(m_j, m_{j'}, M_j)$ when $j' = j_1$ is encountered and proceed by choosing a maximum tight element; this choice will no longer depend on the changing interval $[m_{j_1}, M_{j_1}]$.



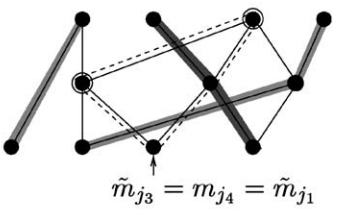
(a) The poset is initially covered by five (shaded) intervals j', j_1, j_2, j_3 and one unnamed. Except for j_3 they are all chains. Steps when $u_i^{(t)}$ change are indicated for the numbered intervals: $u_{j_1}^{(t)}$ for $t = 1$, $u_{j_2}^{(t)}$ for $t = 2$ and $u_{j_3}^{(t)}$ for $t = 2$. We have $t^* = 3$; notice Procedure REDUCE proceeds by selecting $j_4 = j_1$.



(b) Step 1 of Procedure REDUCE. The dashed interval changes; affected $u_{j_s}^{(t)}$ and $u_{j_{s+1}}^{(t)}$ are circled.



(c) Step 2. The previously changed interval is dark shaded.



(d) The new, optimal cover of four intervals. Notice $j_4 = j_1$; since m_{j_1} has changed to $\tilde{m}_{j_1}$, the step affects this new element.

Fig. 2. An example of a poset with two pairs of non-independent incomparable pairs y_1–y_2 and y_2–$u^{(1)}_{j_1}$. The algorithm for this example encounters the difficulty that after setting $\tilde{m}_{j_1} = m_{j_2}$ in Procedure REDUCE, we have $u(m_j, m_{j_1}, M_j) = y_1$ different from $u(m_j, m_{j_2}, M_j) = y_2$. Since neither y_1 nor y_2 are tight, $u^{(2)}_{j_3}$ remains however identical. Two dark shaded chains of subfigure (a) must contain y_1 and y_2 by the strong interval property. (a) The poset is initially covered by five (shaded) intervals j', j_1, j_2, j_3 and one unnamed. Except for j_3 they are all chains. Steps when $u_i^{(t)}$ change are indicated for the numbered intervals: $u_{j_1}^{(t)}$ for $t = 1$, $u_{j_2}^{(t)}$ for $t = 2$ and $u_{j_3}^{(t)}$ for $t = 2$. We have $t^* = 3$; notice Procedure REDUCE proceeds by selecting $j_4 = j_1$. (b) Step 1 of Procedure REDUCE. The dashed interval changes; affected $u_{j_s}^{(t)}$ and $u_{j_{s+1}}^{(t)}$ are circled. (c) Step 2. The previously changed interval is dark shaded. (d) The new, optimal cover of four intervals. Notice $j_4 = j_1$; since m_{j_1} has changed to $\tilde{m}_{j_1}$, the step affects this new element.

We begin with case 1. The proof follows from the counterpart of Lemma 2.9 and Lemma 2.10 below. The proofs need only slight modification compared to their counterparts in Section 2. From here on we use the simplified $u_{jj'}$ notation with reference always to the initial input $\mathscr{I}$. □

Lemma 3.14. *If* $U_{j_1j_2} \leqslant u_{j_1j'}$, *then* $u(m_{j_2}, m_{j'}, M_{j_1}) = u(m_{j_1}, m_{j'}, M_{j_1})$.

Proof. Since $u_{j_1j'} \in [m_{j_2}, M_{j_1}]$ by $U_{j_1j_2} \leqslant u_{j_1j'}$ and $m_{j'} \not\leqslant u_{j_1j'}$, we only have to show that $u_{j_1j'}$ meets the maximality requirement for the definition of $u(m_{j_2}, m_{j'}, M_{j_1})$, that is for all $u_{j_1j'} \leqslant x \leq M_{j_1}$ we have $x \geqslant m_{j_2}$. And this follows by $m_{j_2} \leq U_{j_1j_2} \leqslant u_{j_1j'} \leqslant x$. □

Lemma 3.15. *Let* $u_{j_1}^{(t^*)} \wedge u_{j_2}^{(t^*)} \geqslant m_{j_2}$ *and for some* $j \leqslant k$ and $t \leqslant t^*$ *let* $u_{j_1}^{(t)} \leqslant u_{j_1j}$. *Then* $U_{j_1j_2} \leqslant u_{j_1j}$.

Proof. We are done by

$$U_{j_1j_2} \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t)} \leqslant u_{j_1j},$$

where the first inequality follows by $m_{j_2} \leqslant u_{j_1}^{(t^*)} \wedge u_{j_2}^{(t^*)} \leqslant u_{j_1}^{(t^*)}$ and the second by $t^* \geqslant t$. □

We complete the proof of Theorem 3.13 by considering case 2. We aim to follow the line of Lemmas 2.12 and 2.13 as in the Dilworth problem. However, there the proof is much simpler since we do not have the case of $u(m_j, m_{j_1}, M_j) \neq u(m_j, m_{j_2}, M_j)$, i.e. where the notion u_{jj_1} refers to different elements for inputs $\mathscr{I}$ or $\tilde{\mathscr{I}}$. This implies a weaker result Lemma 3.16 instead of Lemma 2.12 in that we basically replace "$<$" by "$\not\geqslant$"; this lemma is used within the proof.

The main difficulty arises in Lemmas 3.17 and 3.18 corresponding to Lemma 2.13. In the interval cover problem we cannot prove the required inequality $U_{jj_1} \geqslant U_{j_1j_2}$. Instead we prove that, roughly speaking, all elements between the previous and new value of $u_{jj'}$ are tight. We prove this separately for the $u(m_j, m_{j_1}, m_j)$ arising in Procedure REDUCE and in the maximum tight selection steps of Procedure PUSH-TO-TIGHT. Recall that the simplified notation u_{jj_1} is always used with reference to $\mathscr{I}$.

Lemma 3.16. *Assume that* U_{j_1j} *exists, furthermore* $u_j^{(t)} \leqslant u_{jj_1}$ *and* $u_j^{(t)} \leqslant M_{j_1}$. *Then for* $t^* > t$ *we have* $u_{j_1}^{(t^*)} \not\geqslant U_{j_1j}$.

Proof. By contradiction assume $u_{j_1}^{(t^*)} \geqslant U_{j_1j}$. Then we have $m_j \leqslant U_{j_1j} \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t)}$ where the last inequality follows by $t^* > t$. By $u_j^{(t)} \leqslant M_{j_1}$ and $m_j \leqslant u_{j_1}^{(t)}$ the definition of Procedure PUSHDOWN implies $U_{j_1j} \not\leqslant u_{j_1}^{(t+1)}$. We reached a contradiction with the assumption $U_{j_1j} \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t+1)}$. □

Lemma 3.17. *Assume that* $m_j \leqslant y \leqslant M_j$ *is an intermediate value of the intersection* $\wedge$ *in Procedure* PUSH-TO-TIGHT *that satisfies* $m_{j_1} \leqslant y$ *and* $u_{j_1}^{(t)} \vee y \leqslant M_j$. *Assume*

furthermore that j_1 and j_2 *satisfy the condition for Procedure* REDUCE, *i.e.* $U_{j_1j_2} \leqslant u_{j_1}^{(t^*)}$ *for some* $t^* \geqslant t$ *and there is no tight* u *with* $m_{j_1} \leqslant u \leq u_{j_1j_2} \wedge u_{j_1}^{(t^*)}$. *Then*

(i) *for all* $u_{jj_1} \wedge y < x \leqslant y$ *either* $m_{j_2} \leqslant x$ *or* $m_{j'} \leqslant x \leqslant M_{j'}$, *and*
(ii) *if* u_{jj_2} *exists, then for all* $u_{jj_2} \wedge y < x \leqslant y$ *either* $m_{j_1} \leqslant x$ *or* $m_{j'} \leqslant x \leqslant M_{j'}$,

where $j' \neq j, j_1$.

Proof. As in Procedure REDUCE, let

$$u = \min\{m_{j_1} \leqslant u \leqslant u_{j_1}^{(t^*)} : u \text{ is tight}\}.$$

Element u exists since $u_{j_1}^{(t^*)}$ is tight. We have $u \in [m_{j_1}, M_j]$ since

$$m_{j_1} \leqslant u \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t)} \leqslant u_{j_1}^{(t)} \vee y \leqslant M_j.$$

We prove $u \geqslant U_{j_1j_2}$. Since there is no tight element below $u_{j_1j_2} \wedge u_{j_1}^{(t^*)}$, we have $u \not\leqslant u_{j_1j_2} \wedge u_{j_1}^{(t^*)}$. By $u \leqslant u_{j_1}^{(t^*)}$ this implies $u \not\leqslant u_{j_1j_2}$. Thus $u \vee u_{j_1j_2} > u_{j_1j_2}$ and we get $u \geqslant m_{j_2}$ by combining the definition of $u_{j_1j_2}$ and the strong interval property (9) for chain $[m_{j_2}, M_{j_1}]$.

Now we show $u_{jj_1} \wedge y < x \leqslant y$ implies $m_{j_1} \leqslant x \leq M_j$; by the exact same steps we can also prove $u_{jj_2} \wedge y < x \leqslant y$ implies $m_{j_2} \leqslant x \leqslant M_j$. The $x \leqslant M_j$ part is obvious in both cases by $x \leqslant y \leqslant M_j$. And by $u_{jj_1} \wedge y < x \leqslant y$ we get $u_{jj_1} \not\geqslant x$ and thus $u_{jj_1} < u_{jj_1} \vee x$. The inequality $m_{j_1} \leqslant x$ follows by combining the definition of u_{jj_1} and the strong interval property (9) for chain $[m_{j_1}, M_j]$.

Now we assume $x \not\geqslant m_{j_2}$ and thus $x \not\geqslant U_{j_1j_2}$. We use the fact that x and u are dependent; when proving (i), a common interval is $[m_{j_1}, M_j]$ and when proving (ii), it is $[m_{j_2}, M_j]$. Since $x \geqslant u \geqslant U_{j_1j_2}$ is excluded, $u \wedge x < u$ exists; it is not tight by the definition of u, i.e. $m_{j'} \leqslant u \wedge x \leqslant M_{j'}$ for some $j' \neq j_1$. Since u is tight and is contained by the unique interval $[m_{j_1}, M_{j_1}]$, the strong interval property (9) for chain $[m_{j'}, M_{j'}]$ implies $m_{j'} \leqslant x \leqslant M_{j'}$.

We are done by showing j' as above is different from the trivial interval $[m_j, M_j]$ containing x. We prove this by contradiction. Since $m_{j'} \leqslant u \wedge x$ and $j = j'$, we get

$$m_j \leqslant u \wedge x \leqslant u \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t^*)} \vee y \leqslant M_j.$$

However this contradicts that u is tight and thus cannot be contained by $[m_j, M_j]$. □

Lemma 3.18. *Assume that* $m_j \leqslant y \leqslant M_j$ *is an intermediate value of the intersection* $\wedge$ *in Procedure* PUSHDOWN *that satisfies* $m_{j_1} \leqslant y$ *and* $u_{j_1}^{(t)} \vee y \leqslant M_{j_1}$. *Assume furthermore that* j_1 *and* j_2 *satisfy the condition for Procedure* REDUCE, *i.e.* $U_{j_1j_2} \leqslant u_{j_1}^{(t^*)}$ *for some* $t^* \geqslant t$ *such that there is no tight* u *with* $m_{j_1} \leqslant u \leqslant u_{j_1j_2} \wedge u_{j_1}^{(t^*)}$. *Then*

(i) *for all* $u_{jj_1} \wedge y < x \leqslant y$ *either* $m_{j_2} \leqslant x$ *or* $m_{j'} \leqslant x \leqslant M_{j'}$, *and*
(ii) *if* u_{jj_2} *exists, then for all* $u_{jj_2} \wedge y < x \leqslant y$ *either* $m_{j_1} \leqslant x$ *or* $m_{j'} \leqslant x \leqslant M_{j'}$,

where $j' \neq j, j_1$.

Proof. As in Procedure REDUCE, let

$$u = \min\{m_{j_1} \leqslant u \leqslant u_{j_1}^{(t^*)} : u \text{ is tight}\}.$$

Element u exists since $u_{j_1}^{(t^*)}$ is tight; clearly $u \in [m_{j_1}, M_{j_1}]$. We prove $u \geqslant U_{j_1 j_2}$. Since there is no tight element below $u_{j_1 j_2} \wedge u_{j_1}^{(t^*)}$, we have $u \not\leqslant u_{j_1 j_2} \wedge u_{j_1}^{(t^*)}$. By $u \leqslant u_{j_1}^{(t^*)}$ this implies $u \not\leqslant u_{j_1 j_2}$. Thus $u \vee u_{j_1 j_2} > u_{j_1 j_2}$ and we get $u \geqslant m_{j_2}$ by combining the definition of $u_{j_1 j_2}$ and the strong interval property (9) for chain $[m_{j_2}, M_{j_1}]$.

Now we show $u_{jj_1} \wedge y < x \leqslant y$ implies $m_{j_1} \leqslant x \leq M_{j_1}$; by the exact same steps we can also prove $u_{jj_2} \wedge y < x \leqslant y$ implies $m_{j_2} \leqslant x \leqslant M_{j_1}$. We immediately have $x \leqslant y \leqslant y \vee u_{j_1}^{(t)} \leqslant M_{j_1}$. And by $u_{jj_1} \wedge y < x \leqslant y$ we get $u_{jj_1} \not\geqslant x$ and thus $u_{jj_1} < u_{jj_1} \vee x$. The inequality $m_{j_1} \leqslant x$ follows by combining the definition of u_{jj_1} and the strong interval property (9) for chain $[m_{j_1}, M_j]$.

Now we assume $x \not\geqslant m_{j_2}$ and thus $x \not\geqslant U_{j_1 j_2}$. We use the fact that x and u are dependent; when proving (i), a common interval is $[m_{j_1}, M_{j_1}]$ and when proving (ii), it is $[m_{j_2}, M_{j_1}]$. Since $x \geqslant u \geqslant U_{j_1 j_2}$ is excluded, $u \wedge x < u$ exists; it is not tight by the definition of u, i.e. $m_{j'} \leqslant u \wedge x \leqslant M_{j'}$ for some $j' \neq j_1$. Since u is tight and is contained by the unique interval $[m_{j_1}, M_{j_1}]$, the strong interval property (9) for chain $[m_{j'}, M_{j'}]$ implies $m_{j'} \leqslant x \leqslant M_{j'}$.

We are done by showing that j' above has $j' \neq j$. So assume by contradiction that $m_j \leqslant u \wedge x \leqslant u$. This implies

$$m_j \leqslant u \leqslant u_{j_1}^{(t^*)} \leqslant u_{j_1}^{(t)}$$

whence the existence of $U_{j_1 j}$ follows. By Lemma 3.16 and the definition of $U_{j_1 j}$ we get $m_j \not\leqslant u_{j_1}^{(t^*)}$. This contradicts $m_j \leqslant u \leqslant u_{j_1}^{(t^*)}$. □

Now we complete the proof of Theorem 3.13. The steps can be followed on Fig. 2. To define $u_j^{(t)}$ we first run Procedure PUSHDOWN and then Procedure PUSH-TO-TIGHT. We abort this two-step procedure with value y before u_{jj_1} is encountered, i.e. take the intersection $\wedge$ of all current $u_{jj'}$ except u_{jj_1} and return the results. More precisely we defined *two* values y_1 for input $\mathscr{I}$ and y_2 for $\tilde{\mathscr{I}}$. By Lemma 3.11 $u_j^{(t)}$ and the possibly different $\tilde{u}_j^{(t)}$ are equal to the minimum tight elements below

$$y_1 \wedge u(m_j, m_{j_1}, M_j) \quad \text{for input } \mathscr{I}, \text{ and}$$

$$y_2 \wedge u(m_j, m_{j_2}, M_j) \quad \text{for input } \tilde{\mathscr{I}}$$

respectively, or in the latter case, below y_2 itself if $u(m_j, m_{j_2}, M_j)$ does not exist. Recall by Lemma 3.11 that the modification of the intervals does not affect the notion of tightness since the only modified interval $[m_{j_1}, M_{j_1}]$ may no longer play a role after $u(m_j, m_{j_1}, M_j)$ and $u(m_j, m_{j_2}, M_j)$ have been taken into account. We prove $u_j^{(t)} = \tilde{u}_j^{(t)}$.

First we show $u_j^{(t)} \geqslant \tilde{u}_j^{(t)}$ by contradiction. First we derive $y_1 \not\geqslant \tilde{u}_j^{(t)}$. This follows since both $u_j^{(t)}$ and $\tilde{u}_j^{(t)}$ are tight and thus so is their union, implying $u_j^{(t)} \vee \tilde{u}_j^{(t)} > u_j^{(t)}$ cannot be below y_1. Next by $\tilde{u}_j^{(t)} \leqslant y_2 \leqslant y$ and $y_1 = y \wedge u(m_j, m_{j_1}, M_j)$, Lemma 3.6 implies that for all $z \in [m_j, M_j]$ with $z \not\leqslant y_1$ we have $m_{j_1} \leqslant z$. In particular this applies

to $y_1 \not\geqslant \tilde{u}_j^{(t)}$, showing $\tilde{u}_j^{(t)} \geqslant m_{j_1}$. However, then depending on the input condition for j_1 in Procedures PUSHDOWN or PUSH-TO-TIGHT we may apply (i) of Lemmas 3.17 or 3.18, respectively, for $y = \tilde{u}_j^{(t)}$ and $u(m_j, m_{j_1}, M_j) = u_{jj_1}$ to reach the contradiction that either $\tilde{u}_j^{(t)}$ is not tight or it is above m_{j_2} and thus $\tilde{u}_j^{(t)} \not\leqslant y_2$.

To complete the proof we also need $u_j^{(t)} \leqslant \tilde{u}_j^{(t)}$. This is immediate if $u(m_j, m_{j_2}, M_j)$ does not exists and $y_2 = y \geqslant y_1$. Otherwise, we may repeat the same steps as before by exchanging the role of j_1 and j_2 and by applying (ii) of Lemmas 3.17 and 3.18 with $(m_j, m_{j_2}, M_j) = u_{jj_2}$. □

From Theorem 3.13 one easily deduces the main Theorem 3.4. We run the algorithm as long as it either reduces the size of the cover $\mathscr{I}$ by one or outputs a pairwise independent system of one element from each interval. In this latter case the sizes of both the cover and the independent system are optimal. The algorithm terminates in a number of iterations not exceeding the size of the initial cover $\mathscr{I}$; this is bounded either by the total number of intervals or the product of the number of minimal and maximal poset elements.

We remark that the algorithm relies on the choice of tight $u_j^{(t)}$ since by Lemmas 3.17 and 3.18 we may ensure the $u_j^{(t)}$ remain identical when run with the initial or the modified interval system only because of the tightness. The reason that traditional Dilworth proofs are much simpler than presented here are due to the fact that the Dilworth counterpart Lemma 2.12 of Lemmas 3.17 and 3.18 gives the stronger result of $U_{jj_1} \geqslant U_{j_1 j_2}$.

4. Conclusion

We have given a combinatorial algorithm for covering posets satisfying a special property by the minimal number of intervals of the poset. As noticed by Frank and Jordán [10], the result can be applied for certain directed edge augmentation problems.

Our algorithm is independent of Frank [9] and use a completely different approach. The relation of the two algorithms remains open; in fact, it appears even hard to formalize Frank's algorithm in the terminology of the posets of Section 3. We also gave an algorithm for Dilworth's problem; however, that algorithm only slightly differs from the algorithm unwound from the bipartite matching reduction as described in [5] and its the sole purpose is to illustrate and motivate the techniques of the paper.

The main question that remains open is to give a combinatorial algorithm for covering a crossing supermodular weight or demand function p over the poset $\mathscr{P}$. Notice that the min-max result and the non-combinatorial algorithm of Frank and Jordán [10] applies to this case as well. The main reason why the current algorithm fails comes from Lemmas 3.17 and 3.18 where we cannot use $x \wedge u$ is tight since it may have demand zero. In general we may have certain poset elements x and u with, say, $p(x) + p(u) = p(x \vee u)$ and $p(x \wedge u) = 0$ or vice versa. In such a case we may construct examples that Algorithm PUSHDOWN–REDUCE gets stuck with a non-independent system, yet no element $u_i^{(t)}$ can be replaced by a smaller $u_i^{(t+1)}$ in Procedure PUSHDOWN. In this case we apparently we need another step that exchanges the two endpoints of intervals

covering x and u without affecting any other intervals. The correctness of such an algorithm, however, remains open.

Another question that arises is the generalizational power of the interval covering problem. We saw that two algorithmically equivalent problems, Dilworth's chain cover and bipartite matching, are special cases of interval covers; our algorithm generalizes the standard augmenting path matching algorithm. One may ask whether the network flow problem as different algorithmic generalization of matchings could also fit into our framework. Or, extending the question of [19], can we at least tell the hierarchy of hardness of the interval cover, Dilworth, (bipartite) matching and maximum flow problems?

Finally one may be interested in the efficiency of our algorithm for the particular problems that can be handled. Here particular implementations and good oracle choices are needed. One might also be able to give improvements in the sense of the Hopcroft–Karp matching algorithm [17].

Acknowledgements

To the anonymous referee for several rounds of careful reading and suggestions that significantly improved the presentation.

References

[1] A.A. Benczúr, Parallel and fast sequential algorithms for undirected edge connectivity augmentation, Math. Prog. B 84(3) (1999) 595–640; Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 658–667.

[2] A.A. Benczúr, J. Förster, Z. Király, Dilworth's theorem and its application for path systems of a cycle—implementation and analysis. Proceedings of the European Symposium Algebra, Springer, Lecture Notes in Computer Science, Vol. 1643, 1999, pp. 598–509.

[3] A.A. Benczúr, D.R. Karger, Augmenting undirected edge-connectivity in $\tilde{O}(n^2)$ time J. Alg. 37(1), (2000) 2–36; Proceedings of the Ninth ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 500–509.

[4] G.-P. Cai, Y.-G. Sun, The minimum augmentation of any graph to a k-edge-connected graph, Networks 19 (1989) 151–172.

[5] A. Frank, Combinatorial algorithms, algorithmic proofs, Doctoral Thesis, Budapest, Eötvös University, 1976 (in Hungarian).

[6] A. Frank, Augmenting graphs to meet edge connectivity requirements, SIAM J. Discr. Math. 5(1) (1992), 25–53; Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, 1990.

[7] A. Frank, Finding minimum generators of path systems, J. Combin. Theory B 75 (1999) 237–244.

[8] A. Frank, Finding minimum weighted generators of a path system, in: R.L. Graham, J. Kratochvil, J. Nesetril, F.S. Roberts (Eds.), Contemporary Trends in Discrete Mathematics, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, Providence, Rhode Island, Vol. 49, 1999, pp. 129–138.

[9] A. Frank, Finding minimum edge-coverings of pairs of subsets, EGRES Technical Report Series (2001). Available at http://www.cs.elte.hu/egres/

[10] A. Frank, T. Jordán, Minimal edge-coverings of pairs of sets, J. Combin. Theory B 65 (1995) 73–110.

[11] D.S. Franzblau, D.J. Kleitman, An algorithm for constructing polygons with rectangles, Inform. Control 63 (1984) 164–189.

[12] L.R. Ford, D.R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, NJ, 1962.
[13] H.N. Gabow, Efficient splitting off algorithms for graphs, Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 696–705.
[14] M.X. Goemans, D.P. Williamson, The primal-dual method for approximation algorithms and its application to network design problems, in: D.S. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS Publishing Co, Boston, MA, 1997.
[15] M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (1981) 169–197.
[16] E. Győri, A min–max theorem on intervals, J. Combin. Theory B 37 (1984) 1–9.
[17] J.E. Hopcroft, R.M. Karp, An $n^{5/2}$ algorithm for maximum matching in bipartite graphs, SIAM J. Comp. 2 (1973) 225–231.
[18] T. Jordán, On the optimal vertex connectivity augmentation, J. Combin. Theory Ser. B 63 (1995) 8–20.
[19] D.R. Karger, M.S. Levine, Finding maximum flows in simple undirected graphs is easier than bipartite matching, Proceedings of the 30th ACM Symposium on Theory of Computing, 1998.
[20] D.E. Knuth, Irredundant intervals, ACM J. Experimental Algorithmics 1 (1996) (article1, pp.19).
[21] A. Lubiw, A weighted min–max relation for intervals, J. Combin. Theory B 53 (1991) 151–172.
[22] D. Naor, D. Gusfield, Ch. Martel, A fast algorithm for optimally increasing the edge connectivity, Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, 1990, pp. 698–707.
[23] T. Watanabe, A. Nakamura, Edge connectivity augmentation problems, J. Comput. System Sci. 35 (1987) 96–144.